TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

PENG YAO

DOMINANT CONVOLUTIONAL FEATURE CHANNELS FOR IMAGE SUBCATEGORY CLUSTERING

Master of Science thesis

Examiner: Prof. Joni Kämäräinen
Examiner and topic approved by the
Faculty Council of the Faculty of
Computing and Electrical Engineering
on 6 September 2017

ABSTRACT

PENG YAO: Dominant Convolutional Feature Channels for Image Subcategory Clustering
Tampere University of Technology
Master of Science thesis, 43 pages
September 2017
Master's Degree Programme in Information Technology
Major: Information Technology for Health and Biology
Examiner: Prof. Joni Kämäräinen
Keywords: DPM, CNNs, dominant channel, convolutional feature, subcategory, viewpoints, poses, object detection, image classification

This thesis work aims to study what convolutional neural network actually learn and how can we make use of the convolutional neural network features. It carried out a framework to perform image subcategory clustering in the manner of poses or viewpoints. Our work is based on deep convolutional neural network feature maps and using Fuzzy c-means and K-means for clustering. To evaluate the result, we integrated our work with DPM detector and tested on PASCAL VOC 2007 dataset. The result shows our approach did improve the performance significantly in some of the categories, such as bottle, cat, table, sheep and TV monitor comparing with the original DPM detector.

PREFACE

When I apply master study at Tampere University of Technology, in the motivation letter, I wrote my interest and ambition are to learn and achieve something in Artificial Intelligence. And now, here it is. During the past two years study, I am proud to say I did learn a lot in machine learning and TUT helped me open the door to the new domain. I believed, and am still believing, that AI is the future and the world will change rapidly in the coming years. Although the thesis I wrote will most probably be ignored in the academic world, it still means a lot to myself. At least it shows I can apply my knowledge to do something and I am confident to pursue further study or career in machine learning area.

I would like to thank everyone helped me during the project, especially, Professor Joni Kämäräinen, who provided this research topic and gave me many invaluable advices. And he is not only a kind and passionate supervisor, but also an excellent lecturer. I really appreciate his lecture of 'Introduction to Pattern Recognition and Machine Learning'. I would like to say that is the start point of my whole master study and directly lead to this thesis. His lecture is easy to follow and the atmosphere is more than enjoyable. I still remember he uses the example of detecting elf and human in the dark based on their height to explain the probability estimation. Many thanks again.

At last, many thanks to my parents who understand and support me to study abroad. Sorry for absenting from the last two spring festivals. I promise I will come back in the spring festival this year.

22 August 2017 in Tampere, Finland

Peng Yao

TABLE OF CONTENTS

LIST OF ABBREVIATIONS AND SYMBOLS

DPM	Deformable Parts Model
HOG	Histogram of Oriented Gradient
SVM	Support Vector Machine
MLP	Multi-Layer Perceptron
CNN	Convolutional Neural Network
BoW	Bag of Words

1. INTRODUCTION

Image classification and object detection are two challenging tasks in computer vision. Image classification takes an image as the input and output whether the image contains objects from certain classes (e.g. "cat", "car", "table") and object detection is about locating certain objects in the image precisely. More specifically, the outputs of the image classification and object detection include the confidence of the detection result for each class and also bounding boxes to indicate the locations of the objects. The bounding box is often in the form of [left, top, right, bottom] to specify the left-top and right-bottom of the object in the image.

Nowadays we have already achieved tremendous improvements in many domains of object detection, such as handwriting recognition [22], face detection [18, 20] and pedestrian detection [11, 25, 12], but there is still a long way to go in general visual object recognition [21]. What makes image classification and object detection so hard? The main reason is that there are so many factors we have to overcome, for example, illumination, occlusion, scale, deformation, background clutter, intra-class variation and different viewpoints and poses.

Our work is focusing on viewpoints and poses. The viewpoints of an object in an image describe from which angle the object is observed. For example, a car in an image can be in its side view, front view or rear view. Except for the viewpoints, objects could also be in different poses. For example, even in the same viewpoint, a person can be standing, sitting and squatting. It is easy for the human to recognize the same object from different viewpoints and different poses, but how to train a computational model to achieve the same ability is a challenging task. One solution could be to build a comprehensive model which can learn and understand the abstract concept of an object and then apply it to general cases, like us human do. Unfortunately, there is still no significant breakthrough in cognitive science and artificial intelligent in that direction yet. However, in computer vision, it is possible to design an algorithm to learn certain features from a large number of samples in

the same category, and then perform object detection on the new input images by extracting and analysing the features learned previously. In order to handle different viewpoints and poses, it is a common practice to build a model for each viewpoint and pose. In additional, comparing to increasing the size of training dataset, building a model with richer representational structure and constraints can gain more performance improvement [26]. Therefore, it is essential to split the training data into subcategories for individual model training. We define the images of different viewpoints and poses in the same category as subcategory images. Then, the problem is how to better split the training images into subcategories? The easiest method could be done by human manual annotation. This however, is a time consuming job, especially when there are large sets of training images. The other problem is the human bias which will introduce the gray area where the images can be classified into different subcategories by different people. Thus, a machine learning model is needed to handle the job precisely and automatically.

In machine learning, there are typically three different approaches: supervised learning, unsupervised learning and reinforcement learning. Supervised learning is a way to "teach" the machine with inputs and corresponding desired outputs to achieve a general mapping between inputs and outputs. The teaching process is usually the algorithm parameters tuning process. In the contrast, there is no paris of inputs and desired outputs given in unsupervised learning. Unsupervised learning is about to find the hidden structure in the inputs by its own. Reinforcement learning is a process of interacting with a dynamic environment and making adjustments to perform a certain task. There will be rewards and punishments in each interacting according to the performance and finally lead to the goal.

It is common to use supervised learning in image classification. They typically apply bag-of-words (BoW) [17, 15, 23] pipeline starting with local feature extraction, feature encoding and classifier training. In recent work [5], an subcategory-aware object classification framework is proposed and shows state-of-art performance. In [5], a classifier for each subcategory is trained individually following the BoW pipeline.

In the widely used Deformable Part Models (DPM) detector [7], it uses the aspect ratio of the bounding boxes as a simple indicator of subcategory information. This aspect ratio method works in many still cases, for example, in bicycle category, the width and height ratio of the bounding box on a front view bicycle is usually smaller than of the side view bounding box. However, in general, this is not a good solution

if the object in the image is small, or only part of the object is visible.

We proposed an unsupervised learning approach to cluster images into subcategories using convolutional neural networks (CNNs) features. From the observation of the CNN feature maps, specifically the DeepPyramid [10], we found that, for the objects in the same category, the sensitivity of the bounding box area in different CNN feature channels depends a lot on how the object is presented. In other words, for the objects in the same category, with different poses or viewpoints, the certain CNN feature channels will be fired. If we can find those feature channels, similar to BoW model, we can consider each CNN feature channel as a word and its sensitivity as the occurrence counts to form the feature vectors. The difference is that we can manage to perform clustering only using the most dominant channels rather than all the feature channels. Based on the above idea, we first designed a formula to compute the channel sensitivity against the object as channel score. Then we apply the formula to every channel map. Next step is to mine on the channel scores of all the images to find the dominant channels which are supposed to be only sensitive to certain feature of the target object. Once the dominant channels are ready, we pick the channel score of those dominant channels to form the feature vectors of each image. The last step is to apply the clustering algorithms, such as K-means and Fuzzy c-means, to do the classification work.

The experiments are done in a way to replace the aspect ratio based splitting method with our approach to generate the subcategory training images to initialize and train the DPM detector[7] and test on PASCAL VOC 2007 dataset. Then we train the original DPM detector [7] and test on the same PASCAL VOC 2007 dataset. The result shows our work improves the performance in categories of bottle, cat, table and sheep comparing with the DPM detector [7].

2. METHODS

Image classification and object detection are two different tasks. Image classification is about to classify images based on their visual content. Figure 2.1 gives an example to classify images which contain content of the cat, the bus and the boat.

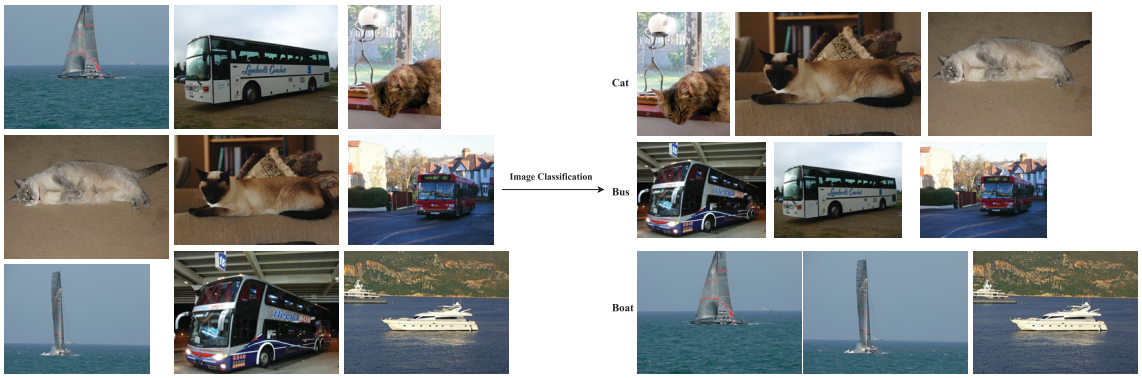

Figure 2.1 *Image classification on categories of cat, bus and boat.*

Object detection is more focusing on the spatial location of the target object in the image. In practice, we often apply them together to not only identify different visual contents in a given image, but also point out their locations. Figure 2.2 gives an example of the bus detection.

There are two popular competitions in the image classification and object detection field, PASCAL Visual Object Classes project (PASCAL VOC) [6] and ImageNet [21]. They have already been widely accepted as benchmarks in image classification and object detection by providing a standard dataset with quality-controlled and human-annotated images and also standard evaluation procedures.

During the past few years, many different systems were proposed to attack those competitions. However, due to the difficulties of processing and understanding image information, none of them reach the performance comparing to human accuracy. In the year of 2008, DPM detector [7] started gaining attention as its outstanding

Figure 2.2 *Example of the bus detection. The input is a RGB image with the size of 375 × 500 × 3. The output is a bounding box in the form of [left, top, right, bottom] to locate the bus in the image.*

performance in PASCAL VOC 2006, 2007, 2008 dataset. Later, deep neural network (DNN) [22] achieved human performance on MNIST [16] benchmark for the first time in the history. After introducing DNN, in the year of 2012, AlexNet [14] won the ImageNet [21] competition by a significant performance improvement against the previous competitors. Since then, computer vision has been improving rapidly and nowadays some systems can even outperform human accuracy. By reviewing the winners in those competitions, Deformable Part Models and Convolutional Neural Networks are the most popular models. It is even proved that DPM detector [7] is actually CNNs in mathematical perspective [10]. Therefore, we further explored the application of CNN features and integrated it with DPM detector [7].

2.1 Deformable Part Models

DPM detector [7] made a huge success in computer vision community and was applied as the core component of many other classification, segmentation and object tracking systems. It is based on HOG [3] feature and latent SVM algorithm [8]. In both theory and practice, deformable parts model is proved to be an excellent approach to build a detector which can adept at rich diverse viewpoints and poses.

2.1.1 Histogram of Oriented Gradient

The underlying of the DPM detector [7] is the HOG feature descriptor. The idea of HOG descriptor is that the distribution of intensity gradient on object local

appearance or edge directions can be successfully used to represent the object in the image. Furthermore, HOG feature descriptor is also able to handle variance in illumination, shadowing, etc., by contrast-normalize local responses.

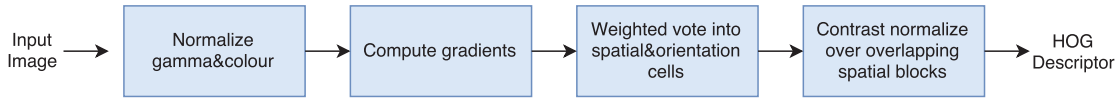

Figure 2.3 *The overview of HOG descriptor extraction.*

The key step to generate the HOG descriptor is image gradient computing. The image gradient is simply a measure of the change in each pixel value along its horizontal and vertical direction neighbours. It contains two concepts: gradient magnitude and gradient orientation. The magnitude indicates how quickly the pixel value changes while the orientation points out in which direction the pixel value changes in 0°-180° ("unsigned" gradient) or 0°-360° ("signed" gradient). Let Δ_x denotes the pixel value changes in horizontal direction and Δ_y in the vertical direction. The gradient magnitude and orientation are defined as,

$$magnitude = \sqrt{\Delta_x^2 + \Delta_y^2} \tag{2.1}$$

$$orientation = arctan\left(\frac{\Delta_y}{\Delta_x}\right) \tag{2.2}$$

From the definition, it is easy to understand that image gradient is invariant to illumination as the Δ value will keep consistent while the neighbouring pixels are changing due to the illumination. Moreover, the image gradient results show that it provides an excellent edge detection (see Figure 2.4). In the practice, as indicated in [3], the simple 1-D centered mask [-1, 0, 1] with no smoothing performs best in image gradient computing.

Once the gradient vectors with both magnitude and orientation are ready, the next step is to divide the image into small local regions called cells and each pixel in the cell votes the weight for orientation histogram bins. Figure 2.5 demonstrates the process of creating cell orientation histogram. In [3], it is proved that the framework works best with 9 orientation histogram bins in 0°-180°, and 16 $\times$ 16 pixel blocks (for normalization) of four 8 $\times$ 8 cells.

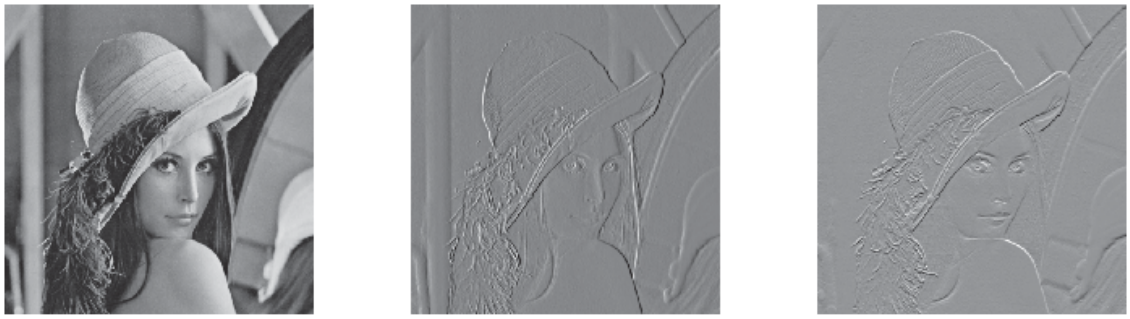

Figure 2.4 *Left one is the original image, middle one is the horizontal direction gradient, right one is the vertical direction gradient.*

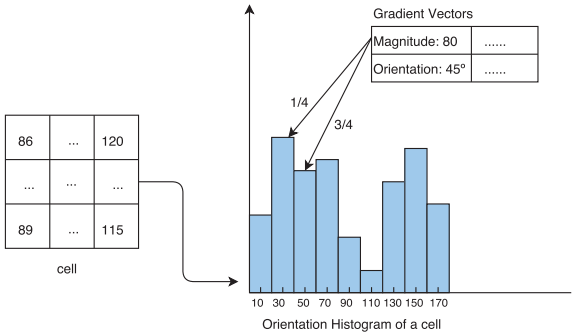

Figure 2.5 *cell histogram of orientation. Given an 8 × 8 cell of an image, the first step is to compute the gradient for each pixel in the cell. Then distribute every gradient magnitude to orientation histogram bins based on their orientations. In this example, the orientation is 45° and the magnitude is 80, so 1/4 of its magnitude contribute to the bin centered at 30° while the other 3/4 of its magnitude belong to the bin centered at 50°. All the gradients in the cell generate the final orientation histogram.*

In DPM detector [7], the dense representation of the image was built in the same way demonstrated in [3], as shown in Figure 2.3.

2.1.2 Deformable Parts and Latent SVM

HOG feature descriptor is a rough representation of an overall structure of an object. In real life, objects often have deformations. In order to take the deformation into account, deformable parts model is employed upon HOG descriptor.

The idea of deformable parts model is that an object can be represented by several parts and each of those parts can be placed in a restricted space region in which

different place has different deformable cost. Furthermore, a corresponding schema is introduced to describe how all the parts can be put together. There is rich work been done in using deformable parts template model in object detection. The DPM detector [7] is based on pictorial structure formulation proposed by [9]. In [9], a classical deformable part model of the human face is demonstrated, in which there are several parts, such as hair, two eyes, nose, mouth and right, left face edges. There are spring linkages between those parts that allow those parts to move flexibly in a certain degree. DPM detector [7] makes use of the idea and construct a star model where one root filter plays as a hub and all the part filters try to find a proper position to connect to it. The root filter is in low resolution used to find an approximate position to cover the whole object. The part filters perform in high resolution to cover small components of the object and link to the root filter.

In the DPM detector training phase, "Latent SVM" was introduced to solve the issue of unknown parts information in the training images. It is a latent discriminative learning process. They first assume the parts locations are the same for all positive examples to train a root filter using standard SVM. Then, find the optimized position for each part based on the deformable cost by fixing the root filter. Then repeat the above process and eventually gain the optimized root filter and parts filters. Figure 2.6 shows the models learned from bicycles which have 3 components and a real detection example.

In order to cover variants of the object from different viewpoints and poses, DPM detector [7] uses mixture models which can be trained by splitting the training images into different appearance-based groups. In practice, DPM detector [7] groups the training images by their bounding box aspect-ratio. More specifically, in the initializing phase, in order to train a model with n components to represent n possible viewpoints and poses of an object, the bounding boxes in the training images of the same category are sorted by their aspect ratio and then split into n groups with equal size. Figure 2.7 demonstrates the training process of the DPM detector [7].

The deformable parts model is considered as the main contribution of DPM detector [7]. However, as explored in [4] that there is only slight performance difference between the use of deformable parts compared to the use of different mixture models. In addition, it is proved that better subcategories can improve the DPM detector [7] performance significantly [5]. Thus, we make use of the DPM detector [7] framework to examine our image subcategory clustering method.

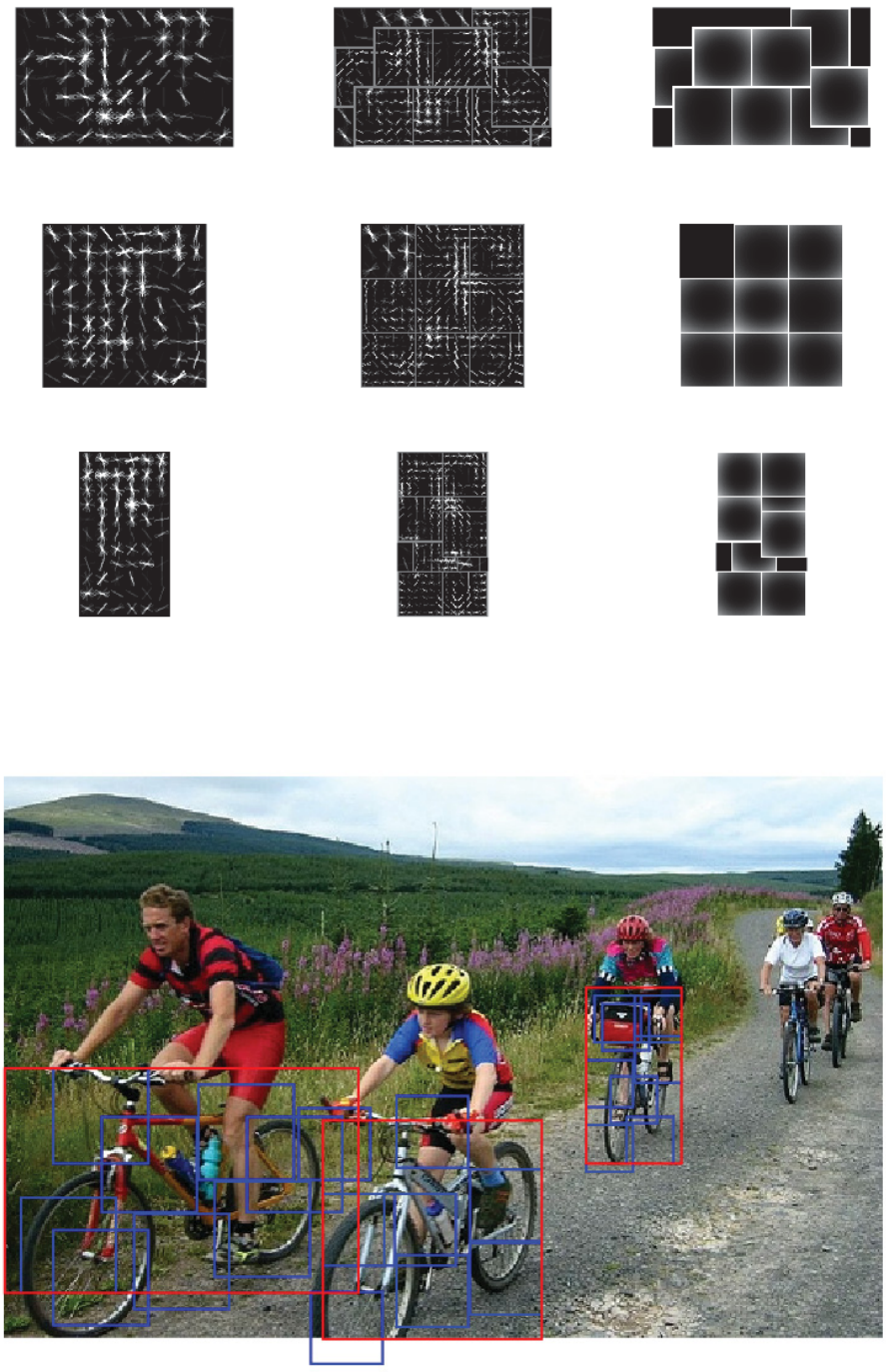

Figure 2.6 *Top: Three models, one in each row, learned from bicycles in the viewpoints of the side, mixed and front. The first one in each row is the root filter in low resolution, the middle one contains several higher resolution part filters and the last one is the model for the spatial location of each part according to the root filter. Bottom: Bicycle detection in the image of real life scene.*

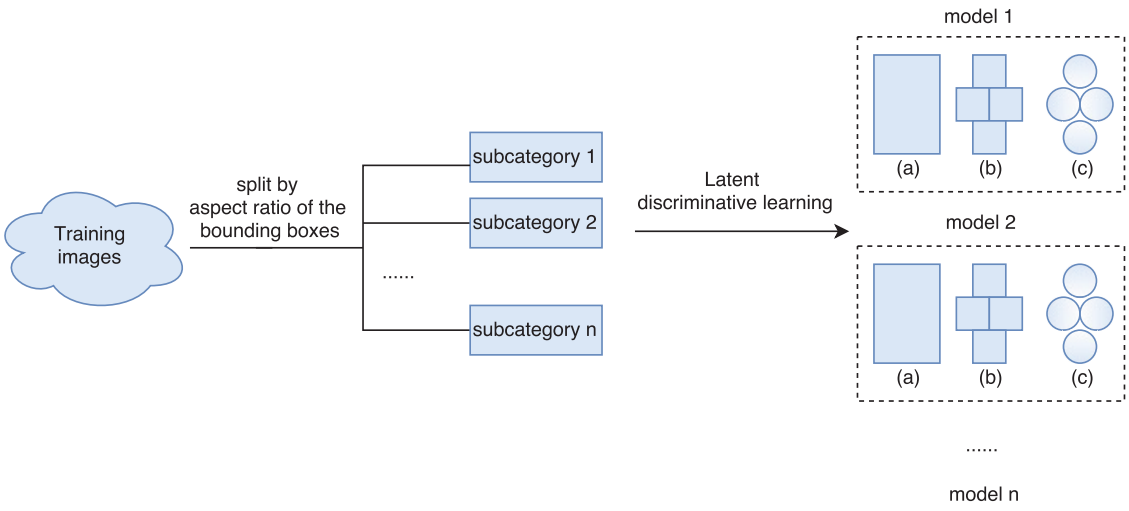

Figure 2.7 *DPM detector using bounding box aspect-ratio to split the training images into subcategories. The training outputs are three parts: a) one root filter in the low resolution, b) several part filters in high resolution and c) the deformable cost for each part filter.*

2.2 Convolutional Neural Networks

Our image subcategory clustering method is based on CNN features. Recently, convolutional neural networks achieved significant performance improvement in image classification and object detection. Comparing to the other traditional methods, CNNs is not a totally new concept but a development from the single perceptron and multi-layer perceptron (MLP). CNNs has a different architecture from MLP by introducing the convolutional layers. The convolutional layers are able to learn the spatial information of the image while the fully-connected MLP are not able to take it into account. Another advantage of CNNs is that it significantly reduces the number of parameters in the network by sharing weights and biases for each of the hidden neurons.

Why CNNs performs so well? As many other object classification and detection systems include feature extraction and classification parts, it is essential to understand the middle feature layers and the last classification layers in CNNs. In [24], it reveals that the features produced inside of the CNNs contain many intuitively understandable characters rather than random, inexplicable patterns. Thus, it is reasonable to use a well-trained CNNs as a feature extractor for other purpose. In our work, we benefit from the visualization of the CNN feature maps and developed the subcategory clustering framework based on it.

2.2.1 Single Layer Perceptron

In order to better understand convolutional neural networks features, it is essential to explore the idea behind it.

As indicated by the name, the neural network is inspired by the nervous system. Our brain has the marvelous ability to process the input information and learn to build relations between them in an unsupervised manner. It is widely acknowledged that the inside mechanism is about to modify the structure and dynamics of the biological nervous system during the learning process. Pavlov's dogs experiment is a classical example of how the relation between bell and food are learned: Pavlov rings the bell when he gives dog food and after many repeats the dog starts salivating when the bell rings without food. So the brain learning process can be considered as an optimisation of the structure and dynamics in nervous system according to the input stimulus and the output response which can be modeled in the similar manner as illustrated in Figure 2.8.

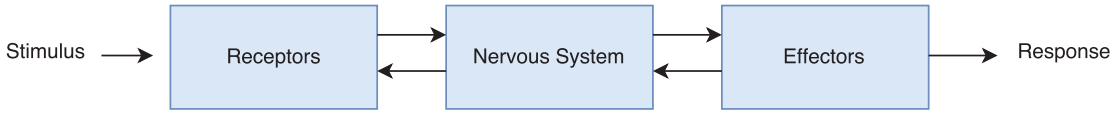

Figure 2.8 *Nervous system continually receive inputs, all kinds of stimulus, process the information by neurons to make decisions, evaluate the result according to the response, then adjust the inside structures and dynamics based on the evaluation and repeats the process.*

Inspired by the biological representation of nervous system, a single perceptron in a mathematic form can be modeled as in Figure 2.9,

where:

$$u = \sum_{j=1}^{n} w_j x_j = W^T * X \tag{2.3}$$

$$y = \phi(u + b) = \phi(W^T * X + b) \tag{2.4}$$

The activation function $\phi()$ is typically the sigmoid function ($\frac{1}{1+e^{-x}}$) as it can take the real-valued inputs and output into a range between 0 and 1. To sum up, in this

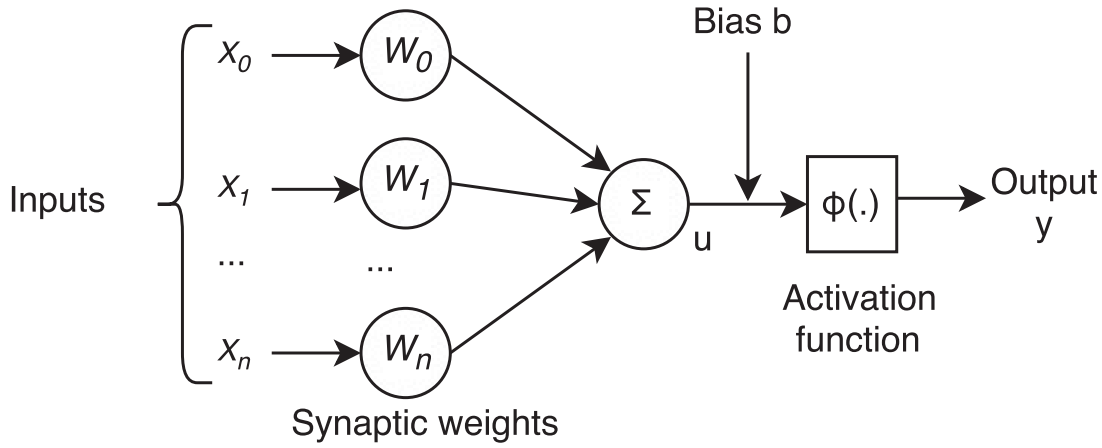

Figure 2.9 *x_0 ... x_n are the input signals, W_n are the synaptic weights which represent the strengths of the connections and can be learned. The sum of all the signals and the weights are feed to an activation function with a bias. The activation function controls the threshold whether the neuron is fired or not, which is the output y (1 or -1, for example).*

model, in order to perform whether a neuron fire or not, we apply dot production with the inputs and their weights and then add a bias to feed an activation function. If the output of the activation function is greater than the defined threshold, then fire the neuron, otherwise remain silent.

We need to train the network to learn correct output against given input, which is all about to adjust the synaptic weights. It starts from a set of random weights. Given amount of training pairs each has the input $X_k = [x_{k,1}, x_{k,2}, ..., x_{k,n}]$ and its desired output y_k^*, the task is to compute the output of 2.4 and then adjust the weights to find the optimised W to minimise the error $|y - y^*|$. For convenience, we define the training error as:

$$E = \frac{1}{2}\sum_{i=1}^{k}(y_i^* - y_i)^2 \tag{2.5}$$

The gradient descent algorithm is used to search the minimum point in the error surface. The initial weights W_{init} are modified by small steps in the direction that generates the steepest decent along the error surface E.

After learning on a large set of training data, a single perceptron model is able to classify an unknown sample into its own group. However, it can only perform on

linearly separable data set and the decision boundary must be hyperplanes.

2.2.2 Multi-Layer Perceptron

In order to deal with more complex cases, a hidden layer, or several hidden layers, is introduced to construct a multi-layer perceptron, as shown in Figure 2.10.

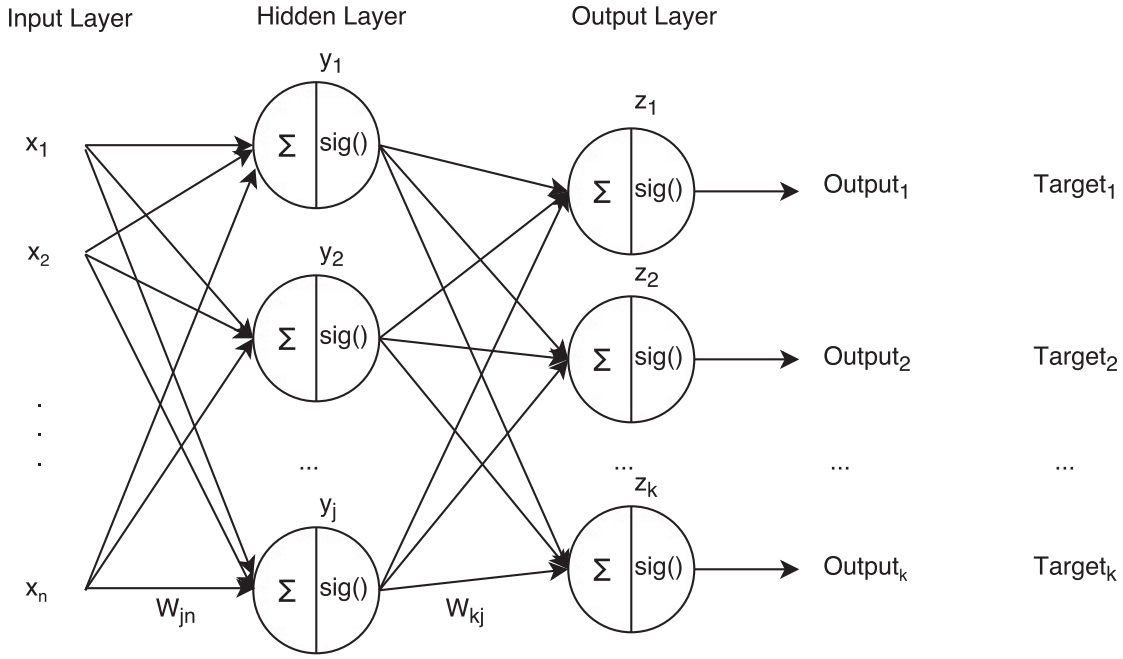

Figure 2.10 *x_0 ... x_n are the input signals, W_{jn} is the weight of the connection between j_{th} neuron in the hidden layer and n_{th} input from input layer (W_{kj} works in the same manner). Sig() is the sigmoid activation function to threshold the output. $Output_k$ are the network outputs, and $Target_k$ are the true labels. The input signals propagates the network layer by layer in a feedforward manner. Each step performs the same as it in single perceptron in 2.9. The difference is that MLP uses back-propagation to learn the weights.*

The mathematic form can be written as:

$$Output = sig(W^T_{hidden} * Output_{hidden} + B_{hidden}) \tag{2.6}$$

or without activation functions but a linear output

$$Output = W^T_{hidden} * Output_{hidden} + B_{hidden} \tag{2.7}$$

where

$$Output_{hidden} = sig(W_{input}^{T} * X + B_{input}) \tag{2.8}$$

and W denotes the weight vector, X is the vector of inputs, B is the bias vector.

Multilayer perceptrons with even only one hidden layer (Figure 2.10), are powerful enough to approximate general mappings from one finite dimensional space to another [13].

Although MLP is the network of single perceptrons, the learning process is more complex. Because there is no target value provided in the hidden layers which makes the error computing is no more straight forward. Taking $Output_1$ in Figure 2.10 as an example, the learning steps in single perceptron can still apply for adjusting weight W_{kj} as we know the $Target_1$ and easy to calculate the error. But for the weights W_{jn}, it is hard to calculate since there is no target value for the hidden output so that there is no direct way to compute the error produced by this layer. The backpropagation algorithm addresses the issue in a way of propagating the error from the last layer backward to the previous one's layer by layer. Thus we don't need to calculate the error at each layer.

MLP has the significant advantage against single perceptron model and can be trained with backpropagation algorithm. The number of hidden layers plays the essential role in MLP. Any lack of success of the network may be caused by the inefficient number of hidden unit [13]. However, with the increasing of hidden layers, the full connection nature of MLP makes the network more and more complex and raises the number of parameters dramatically which slows down the training process significantly. Another issue is that the gradients start vanishing when there are too many hidden layers as the error decreases exponentially with the number of layers in backpropagation.

2.2.3 Convolutional Networks

In MLP, the network treats the input as a vertical line of neurons. In order to take the advantage of the spatial information of the input, instead of full connection in MLP, convolutional networks use region connections with slide window method to generate the hidden layer which is named convolutional layer.

The region a hidden neuron connects to is called "local receptive field". As shown in Figure 2.11, we slide the window with the size of the receptive field one step at each time to cover all input pixels and create a hidden layer (convolutional layer). The size of the moving step is defined as "stride". It is straight forward that the stride parameter is directly related to the size of the convolutional layer. The larger the stride is, the smaller the size of a convolutional layer is.

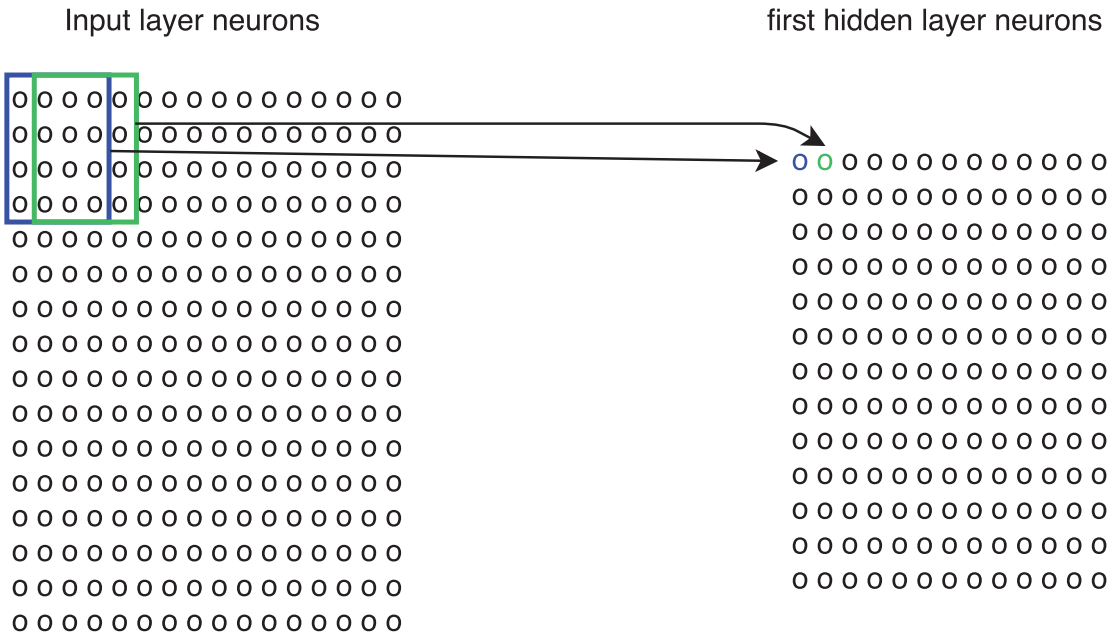

Figure 2.11 *Example of a* 4×4 *Local receptive field for hidden neurons on a* 16×16 *input. The output is a* 13×13 *convolutional layer.*

In every local connection, it is the same as in single perceptron, for each neuron in the convolutional layer, it connects to all the pixels in that region and each connection learns a weight (Figure 2.12). All the weights together is defined as a "filter".

Furthermore, all the neurons in the convolutional layer share the same weights and bias. It means that every single neuron in the following convolutional layer will learn exactly the same feature. For example, if the weights are learned to detect the edge in the receptive field, a convolutional layer neuron will fire if there is edge feature in its connection region regardless its position. Therefore, we can learn different features with many filters. The output of the filter is called "feature map" and we refer the filters as "feature channels". The number of the feature channels is the dimension of a convolutional layer.

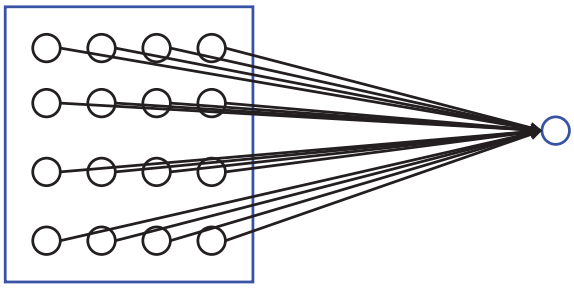

Figure 2.12 *Full connection between a 4×4 local receptive field and a neuron in convolutional layer.*

The other benefit, also the most significant advantage, from sharing weights is that it reduces the number of parameters in the training phase for CNNs. For instance, given an image with size 256×256, with a filter size 8, for CNNs, the number of shared weights is $8 \times 8 = 64$. If we have 128 feature channels, then the total number of parameters in the convolutional layer is $64 \times 128 = 8192$. In comparison, if we feed the same image to a MLP with 30 hidden neurons, the total parameters reach to 1966080 which is 240 times more than the number in CNNs.

Except for the weights, the other parameter and techniques used in CNNs are the depth of the network, zero-padding, pooling layer and fully-connected layer.

Zero-padding can be applied on the input layer to control spatial size of the filter output. The idea is to pad zeros on the border of the input layer to make sure the receptive field size match the input while sliding the window in a certain stride. For example, given an input with size of 5, receptive field size of 3, we can keep the size of output as the same with input by setting zero padding of 1 and stride to 1 (Figure 2.13).

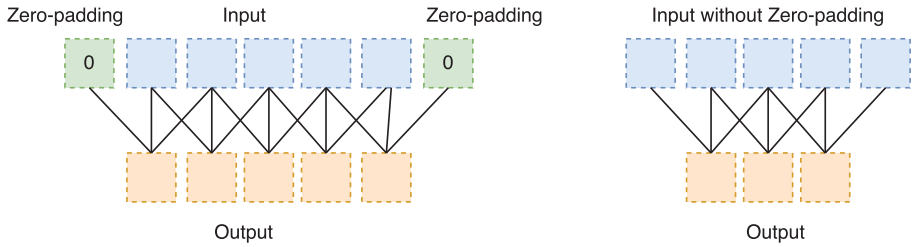

Figure 2.13 *Given an input with size of 5, receptive field size of 3 and stride as 1, Left: keep the output as the same size of input by applying zero-padding. Right: Output size is 3 without zero-padding.*

Let's denote the stride as S, the input size as V, the filter size (window size) as F, the amount of padding zero as P. The size of the convolutional output V', which is the convolutional layer, is:

$$V' = (V - F + 2P)/S + 1 \tag{2.9}$$

Figure 2.12 is an example with no zero-padding and stride one, the convolutional layer size is $V' = (16 - 4 + 2 \times 0)/1 + 1 = 13$. If we want to keep the same spatial size as input, we need to extend the input border with size two and full of zeros. Equation 2.9 is useful to examine the design of the filter size or strides. If V' is not an integer, then it indicates the filter cannot neatly apply to the input.

The depth of the network is the number of convolutional layers of a network which is different from the dimension of a signal convolutional layer. More convolutional layers can be added to the network. The more depth we construct the CNNs, the more intuitively desirable properties the feature maps have [24].

Pooling operation can be considered as a sampling process to reduce the spatial size of the given input. It uses the same slide window approach to go through the whole input with a given filter and stride. The most common filter is the max-pooling which simply chooses the max value in the window area. It is usually introduced right after the convolutional layer. In practice, it is very rare to apply zero-padding for pooling layer.

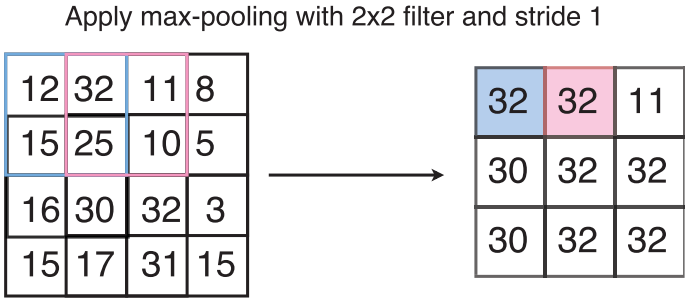

Figure 2.14 *Max-pooling is simply picking the max value in the area covered by slide window.*

Fully-connected layer is identical to the MLP structure and performs in the same way. Every neuron in this layer connects to all the neurons in the pooling layer and

applys dot product and activation function to produce the final output.

Figure 2.15 illustrated a typical CNNs with the network depth of one and dimension of n in the convolutional layer.

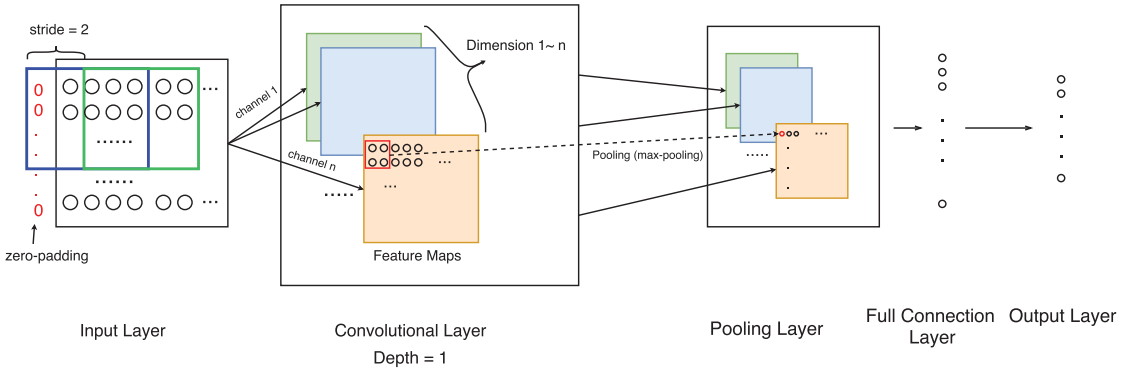

Figure 2.15 *A typical CNNs architecture with the depth of network as* 1*, the dimension of the convolutional layer as* n*, zero-padding, pooling layer, and full connection layer at the end.*

The training process of CNNs is mostly the same with it in MLP using back-propagation, but it needs to be modified in a way to take the local connectivity and max-pooling into account since the CNNs is not the fully-connected network anymore.

Once a trained CNNs is ready, we can extract the feature maps at the certain convolutional layer for further analysis. In our case, we perform image subcategory clustering based on the feature maps extracted from the fifth layer of the AlexNet [14].

2.3 Clustering

In many image classification and object detection competitions [6, 21], there is often no subcategory information available. Therefore it is impractical to perform supervised learning to classify images into subcategories.

We use clustering to reveal the hidden structures from those unlabeled images. The clustering is one of the unsupervised learning algorithms and it is a grouping process of gathering a set of objects which are more similar to the ones in the same set but less similar to those in other groups.

K-means [19] and fuzzy c-means [2] are two widely used clustering algorithms. We apply both of them on CNN features for image subcategory clustering.

2.3.1 K-means

K-means [19] clustering is about to group the elements into K clusters in which every cluster is represented by the mean of the elements in the cluster. Elements are assigned to the cluster with the nearest mean vector (cluster centroids).

Given a dataset with n samples $(x_1, x_2, x_3, ..., x_n)$, each of which has d dimensions, to divide them into k clusters $(C_i, C_2, C_3, ..., C_k)$, the K-means algorithm dedicates to find the centroids which minimise the criterion within the cluster:

$$\underset{C}{argmin} \sum_{i=1}^{k} \sum_{x \in C_i} \|x - u_i\|^2 \tag{2.10}$$

where u_i is the centroid (mean of the points) of the cluster C_i

Figure 2.16 gives a demonstration of the standard K-means algorithm:

1. Assign the number of clusters
2. Initalize the cluster centroids randomly
3. Assign the elements to the nearest centroids
4. Update the centroids of each clusters by averaging all the elements in the cluster
5. repeat step 2 and step 3 until the centroids do not change anymore or reach the given number of iteration

2.3.2 Fuzzy c-means

Fuzzy c-means [2] is similar to K-means, but instead of assign a given element to a cluster exclusively, fuzzy c-means allows the element belonging to more than one

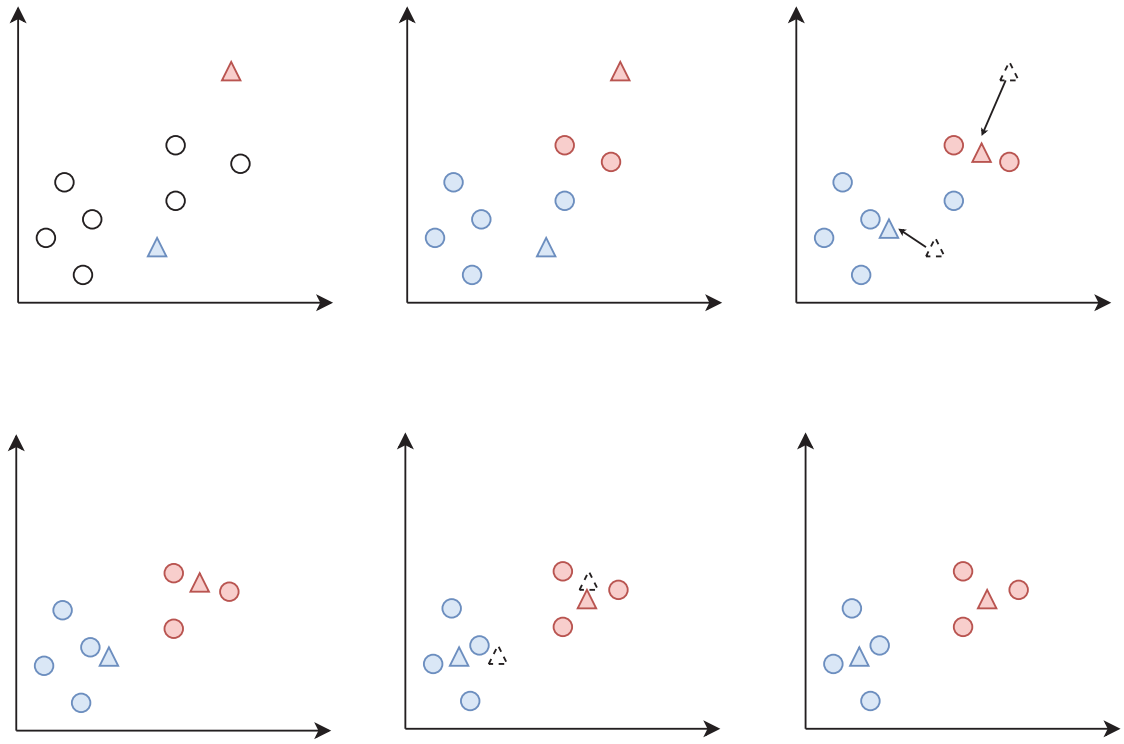

Figure 2.16 *Demonstration of K-means algorithm. Start from two random centroids, blue and red triangle. Then assign elements to nearest centroids to form two clusters. Compute the new centroids by averaging the elements in each cluster. Repeat the process until the centroids stay constant. The last figure is the final clustering result.*

cluster by calculating the degree of the membership in the share clusters rather than simply using the distance to every cluster centroids.

The same with K-means, to cluster n elements in to c clusters, the Fuzzy c-means also aims to minimize a criterion function:

$$\underset{C}{argmin} \sum_{i=1}^{n} \sum_{j=1}^{c} W_{ij}^{m} \|x - u_j\|^2 \tag{2.11}$$

where u_j is the jth cluster centroid, and $W_i j$ is the degree of membership of ith element in jth cluster, which is defined as:

$$W_{ij} = \frac{1}{\sum_{k=1}^{c} (\frac{\|x_i - c_j\|}{\|x_i - c_k\|})^{\frac{2}{m-1}}} \tag{2.12}$$

where m is the fuzzifier parameter, $m \in R$ and $m > 1$. The fuzzifier m is used to tune the fuzziness of the clustering, the larger the m is given, the more fuzzier the clustering is.

The centroid of the cluster is not just simply the mean of all the elements in the cluster as in k-means but take the membership weights into account:

$$C_j = \frac{\sum_{i=1}^{n} w_{ij}^m x_i}{\sum_{i=1}^{n} w_{ij}^m} \tag{2.13}$$

The standard Fuzzy c-means process is also similar to K-means:

1. Assign the number of clusters to perform
2. Randomly assign the membership w_ij for each element in each cluster
3. Update the centroids of each clusters by 2.13
4. For each element, compute the new membership weight w_ij for each cluster
5. repeat step 3 and step 4 until the centroids do not change anymore or reach the given number of iteration, or the maximum of the change in updating w_ij is smaller than a given criterion

Both of the K-means and Fuzzy c-means require to define the number of clusters which is one of the major drawback in some of the data pre-processing. They also have other disadvantages, such as sensitive to outliers, local minimum (the result depends a lot on the initialisation), large memory consumption, etc. But they are simple, fast and perform quite good if the data is well separated, especially if we can estimate the number of clusters before hand.

2.4 Pascal VOC data sets

For comparison, we use the same PASCAL Visual Object Classes Challenge 2007 datasets [6] as in DPM detector [7]. PASCAL VOC challenge had run during the year from 2005 to 2012 and the aim is to recognize objects of various categories in the images from realistic scenes. It is an annual competition focusing on five challenge

tasks of classification, detection, segmentation, action classification, and person layout. Although it finished in the year of 2012, it is still a highly acknowledged data set to practice and exam the performance of different algorithms.

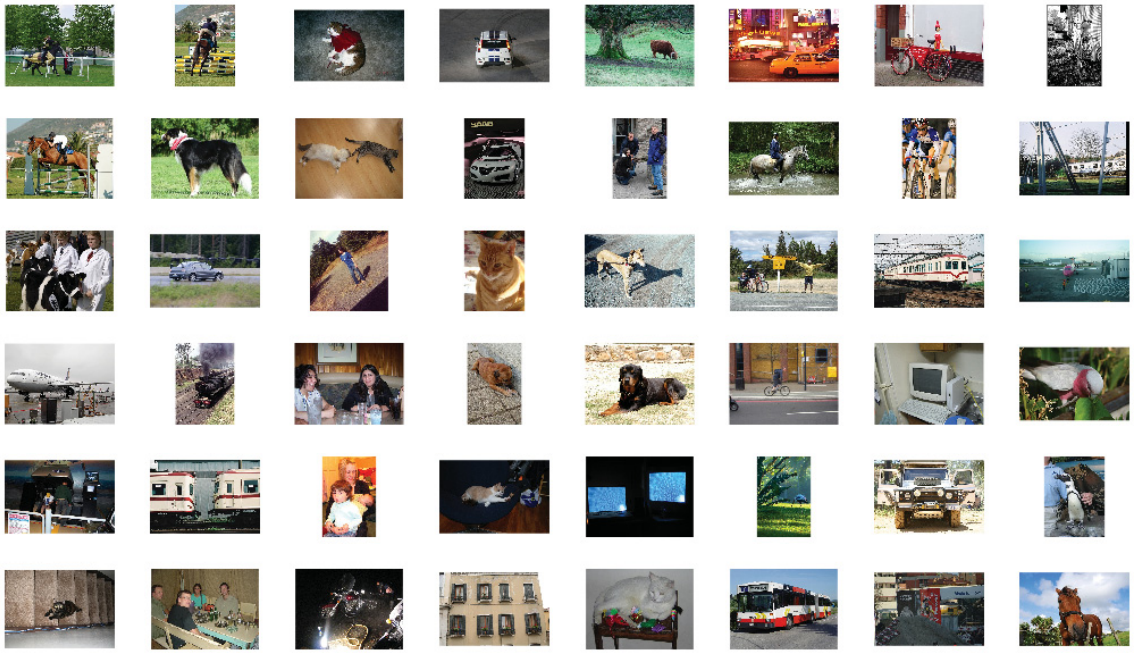

Figure 2.17 *Images randomly chosen from VOC2007 training dataset in random categories.*

PASCAL VOC data sets use rectangular bounding box to annotate the objects in the images and provides the standard development tool kit in MATLAB code to access the annotations and also evaluate the algorithm performance.

Take the PASCAL VOC 2007 as an example. For the classification and detection tasks, 4 data sets are provided: training data, validation data, training and validation data as a whole, test data. In the real competition back then, the annotation file of the test data was not available for the developer but only used for the final result test, which is saved in server side. Nowadays the organizers have already made all the data available to the public which makes it easy for everyone to test their algorithms locally rather than submitting to the evaluation server. The test data is not supposed to use for parameter tuning and we also didn't use it in our training to strictly follow the rule in the competition in order to compare the performance of existing works.

The whole image datasets for classification and detection tasks include 9,963 images,

24,640 annotated objects in 20 categories, such as person, bird, cat, cow, dog, horse, sheep, aeroplane, bicycle, boat, bus, car, motorbike, train, bottle, chair, dining table, potted plant, sofa, tv/monitor. Figure 2.17 gives the overview of how those images look like.

The submission of the results of classification task is supposed to be in separated files for each category. Each line of the file is one record with a confidence value generated by the classifier.

It is similar to the detection task in the submission which also has to be separate files for each class. In addition to the confidence value, a bounding box in the format of $[left, top, right, bottom]$ is required to identify the location of the objects.

For both tasks, the precision/recall curve is used to judge the performance and the principal quantitative measure is the average precision (AP).

3. DOMINANT CNN FEATURE CHANNELS CLUSTERING FRAMEWORK

In this section, we discuss the CNN features and our approach to mine the CNN feature channels.

CNN features have many intuitively desirable properties [24]. Each feature map is generated by a certain channel. Therefore, as illustrated in Figure 2.15, the more dimensions a convolutional layer has, the more channels there are and the more features are learned in that layer. The problem is that in real life images, there are not only target objects but also innumerable backgrounds and other objects. It results that not all the channels learn features from the target objects. In our work, in order to make use of the CNN feature channels for further subcategory clustering, we need to mine all the channels to find out the "dominant CNN feature channels" which are supposed to significantly represent the target objects.

We proposed a framework to extract dominant CNN feature channels by quantizing CNN feature maps in the manner of sensitivity. After extracting the dominant CNN feature channels, both K-means and Fuzzy c-means are applied to achieve the image subcategory clustering.

In particular, the implementation is based on the CNN feature pyramids called DeepPyramid [10]. The underlying of the DeepPyramid [10] is AlexNet [14].

3.1 Pyramid CNN feature channels

The overall architecture of DeepPyramid is depicted in Figure 3.1. The input is a RGB colour image and it will be downsampled to produce a collection of images called "image pyramid" from fine resolution to coarse resolution in a certain scale level. Each pyramid level image is fed to a convolutional neural network. Any convolutional neural network can be introduced to produce the DeepPyramid. The

output is the feature map generated by the last convolutional layer of the network. In our case, we keep the same configuration with DeepPyramid using the AlexNet [14] which ends at convolutional layer 5 (Conv5). Therefore, there are 256 feature channels in the output. In DeepPyramid, the scale level is set to 7, so each feature channel has corresponding 7 scaled feature maps.

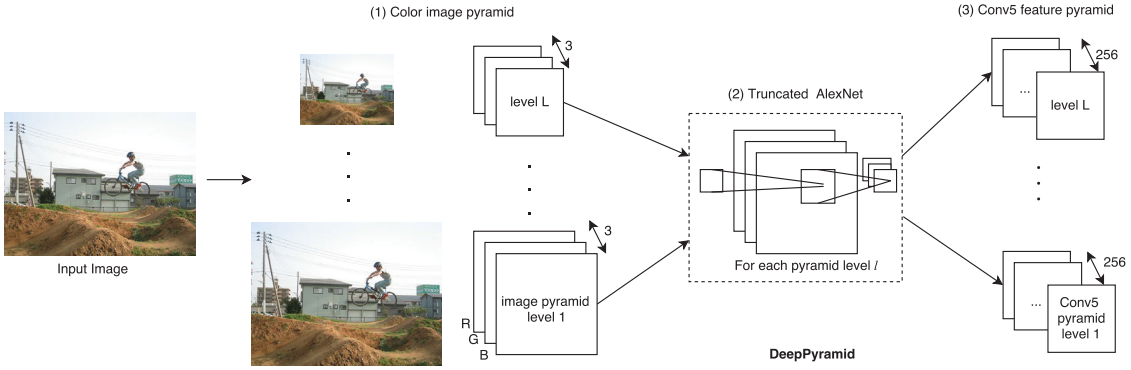

Figure 3.1 *An input RGB colour image is downsampled into an image pyramid. Each image in the pyramid has 3 dimensions (R, G, B) and is fed into AlexNet [14]. The feature maps are extracted at the convolutional layer 5. As the input is an image pyramid, the output is also a Conv5 feature pyramid.*

In AlexNet [14], the network contains 7 hidden layers in which the first 5 layers are convolutional layers and last two are full connected layers. They applied Max-pooling in the first 2 and the last convolutional layers. AlexNet has a fixed input size of $3 \times 244 \times 244$. In other words, it is supposed to deal with the RGB image in size of 244×244.

In order to deal with the different size of the input images, in AlexNet, the input image is first rescaled in the way that keeps the shorter side as the length of 256. Then cropped the center area out by size of 256×256. There are two advantages of this approach. One is that it makes any given image fit into the network input without losing much of the information of the target object. The other is that we can increase the training data by randomly cropping 244×244 size areas in a 256×256 space while keeping the main object in most of the cases. Thus it helps to reduce the overfitting problem.

In more detail, AlexNet uses 11×11 local receptive size with the stride length of 4 and followed by a Max-pooling operation on the region of 3×3 in the first convolutional layer. They defined the dimension of the first layer as 96 which will

produce 96 feature maps accordingly.

After Max-pooling in the first layer, they use 5×5 receptive size, same Max-pooling as in the first layer, but increase the dimension to 256 in the second layer.

The next 3 convolutional layers use the same local receptive size of 3×3 but in different dimensions of 384, 384 and 256 respectively. Therefore we have 256 feature maps after truncating the AlexNet at the fifth layer. Overall there will be $1792 = 7 \times 256$ feature maps, as for each image, there are 7 levels pyramid in different resolutions.

Figure 3.2 illustrates the visualisation of CNN feature maps from all levels of channel 186 and channel 246 on different bicycle images from DeepPyramid. It shows that different neurons may respond to different features in the input. For example, feature channel 246 is more sensitive to the side view of the bicycle, the feature of the wheels, while feature channel 186 is more sensitive to the front view of the bicycle. This observation provides a possible angle to achieve subcategory clustering of poses and viewpoints by digging out the most dominant feature channels.

3.2 CNN feature channel score

In order to find out all the dominant feature channels, we need to quantize the sensitivity of every feature channel regarding the target object. In the feature maps shown in Figure 3.2, the ones in channel 186 with level 5 and 6 in the third row, and the ones in channel 246 with level 6 and 7 in the first row, are the feature maps we are interested, because they only target on the bounding box area and are sensitive in terms of size and intensity. Thus, they should gain higher channel scores while others should yield lower scores. In this example, for the first image with the side view of the bicycle, channel 246 will produce much higher score than channel 186, on the other hand, for images with more front view bicycles, the channel 186 will gain a higher score than channel 246. That is the key discrimination we will use to subcategorize images in different viewpoints.

Based on the above idea, we designed a formula to calculate the channel score with the following variables: channel sensitivity, channel sensitivity size, channel size, object bounding box sensitivity, object bounding box sensitivity size, object bounding box size.

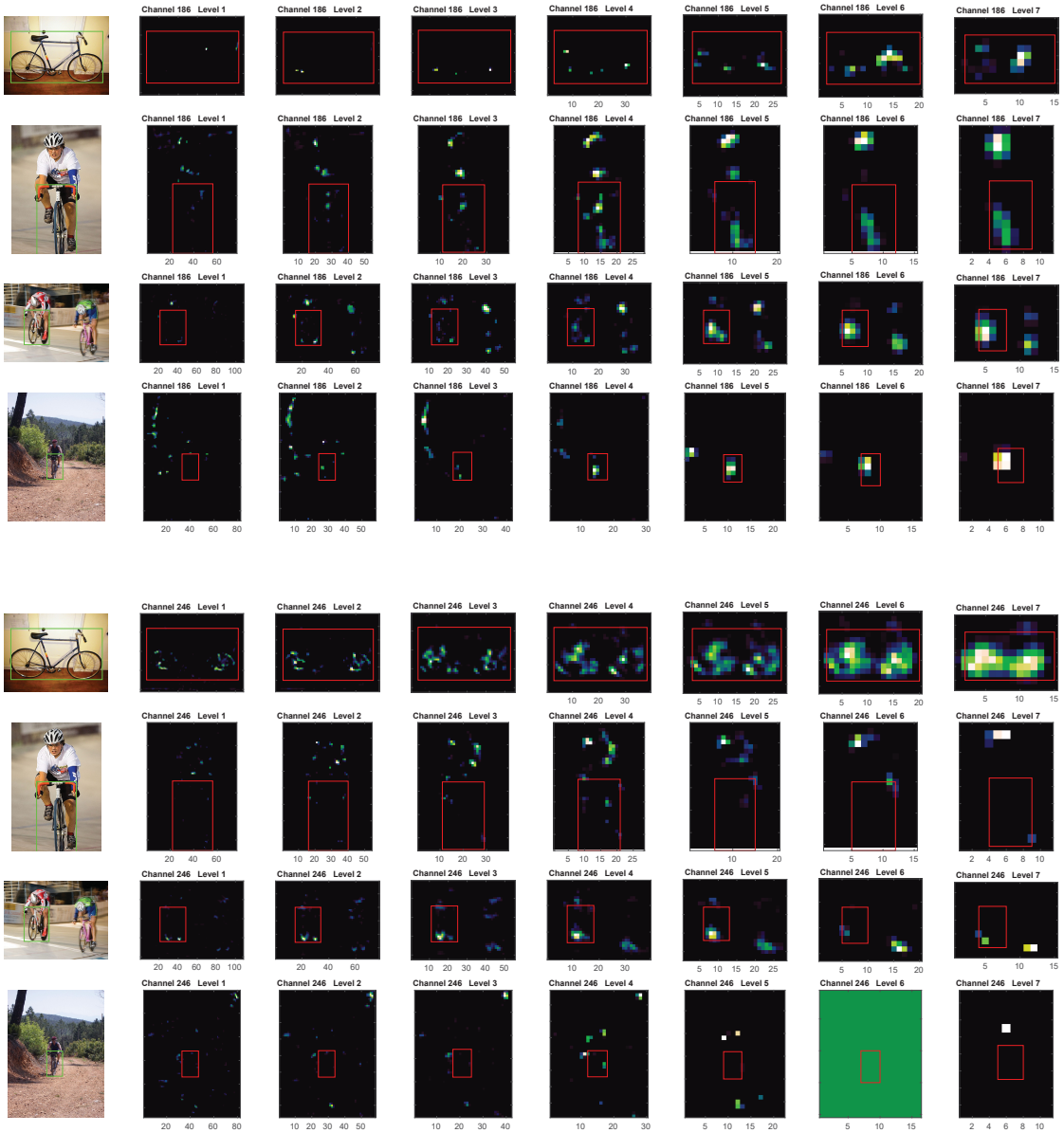

Figure 3.2 *The visualisation of DeepPyramid feature channel 186 and 246 on different bicycle images. The more bright area there is, the higher the channel sensitivity is. It shows clearly that channel 246 is way more sensitive to pure side view of the bicycle than channel 186 while almost insulates to pure front view of the bicycle. It also happens to channel 186 to front view of the bicycle respectively. In row 3 and 7, objects are found in both channels, however the sensitivity is very low in both channels. It can be explained that the view point of the bicycle in this image is from a certain angle between pure side view and pure front view which includes both features and makes both channels fired on it, but with relatively lower sensitivity.*

Channel sensitivity reflects how well this channel response to the input in an overall point of view while the object bounding box sensitivity will give us the sense of how well the target object area is detected. In some channel, it may sensitive not only to some feature of the target object, but also sensitive to other background objects which have the similar feature, such as in Figure 3.2, channel 186 in row 2 which are not only sensitive to bounding box area, but also respond to the helmet area. With the ratio of these two variables, we can understand how much response of the channel located in our interested area.

We use the actual pixel values in feature maps to represent their sensitivity. The higher the value is, the more sensitive the area is. Then it is easy to understand that channel sensitivity is just the sum of all the pixel values in the feature map and object bounding box sensitivity works in the same manner but just in the bounding box area. However, in some cases which have more than one object in the same image, it is essential to remove the effects of other neighbor objects and only focus on each of them individually. For example, in row 3 of Figure 3.2, when we calculate the channel sensitivity for the left bounding box object, it is necessary to mask the other object bounding box area in summing the whole channel sensitivity.

It is not enough to identify the object well responsed channel only from sensitivity perspective. The sensitivity size also plays a very important role in the game. In Figure 3.2, Channel 186 with level 4 in row 4, gives an example in which the object bounding box sensitivity represents majority of the response in the whole channel, but it is not a good candidate for high score, because it has a bright spot in the bounding box, but there is also an even bigger sensitive area for background. To deal with these cases who have few high pixel values in object bounding box, meanwhile there is a large area full of low pixel values, we introduce the sensitivity size variable. An easy way to calculate sensitivity size is to count the number of none-zero values in the area. Therefore, the channel sensitive size is the number of all none-zero values in the whole feature map and object bounding box sensitive size only take the bounding box area into account. We use the same method in sensitivity calculation to deal with the multiple objects cases. So the higher the ratio between object bounding box sensitivity size and channel sensitivity size is, the higher the channel score may achieve.

There is another case, as shown in Figure 3.3. It illustrates the feature map from channel 1 for a side view bicycle. In the feature map of level 1 and level 2, the

bounding box area not only contains the majority of the sensitive area of the whole channel, but also contain the most of the sensitivity. However, it is too sparse to deserve high channel score. To deal with such cases, we need to take the sensitive density into account. It can be done easily by dividing bounding box sensitivity with bounding box sensitivity size.

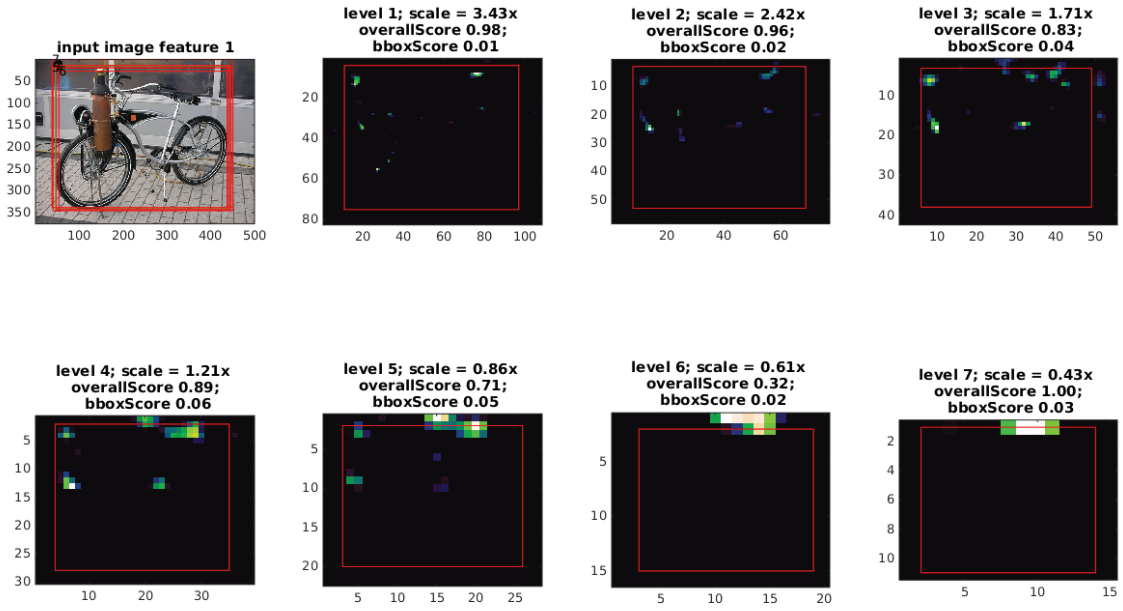

Figure 3.3 *The overallScore indicates the percentage of the pixels falling inside the bounding box. The bboxScore is the pixel density inside the bounding box. As we can see, in the feature maps of level 1 to level 5, although the overall scores are very high due to majority of the pixels lay inside the bounding box, but the pixels are very sparse inside the bounding box. Overall, those feature channel scores should be low.*

With all the aspects discussed above, we can formulate the channel score as 3.1. Where the S is the channel score, SNb and SNc are bounding box sensitivity and the whole channel sensitivity, SSb and SSc are bounding box sensitivity size and channel sensitivity size, Sb is the bounding box size.

Figure 3.4 illustrates the process and structure of channel score vectors of a given image set. For each channel, there are 7 feature maps as there are 7 inputs in different resolutions, in the end, we sum up all of them as the final channel score.

$$S = \frac{SNb}{SNc} \times \frac{SSb}{SSc} \times \frac{SNb}{SSb} \times \frac{SSb}{Sb} = \frac{SNb}{SNc} \times \frac{SNb}{SSc} \times \frac{SSb}{Sb} \tag{3.1}$$

Figure 3.5 shows the feature maps that are sorted by the channel scores. After quantization of sensitivity and sorting the channel scores, it does reveal that there are discriminations in channels on different category images, such as there is the more frequent occurrence of channel 186 in front view bicycles while channel 246 is more often shown in side view bicycles.

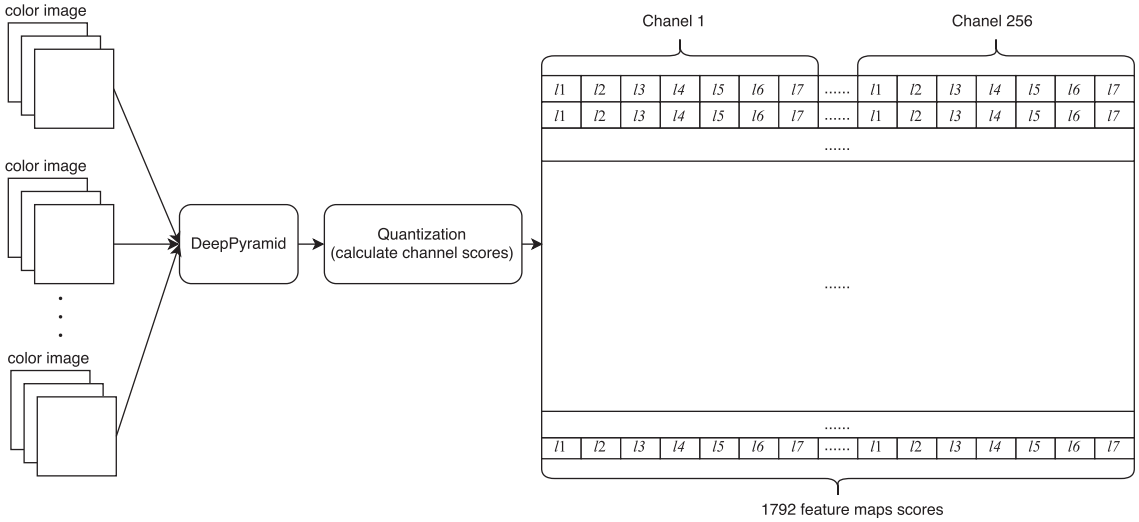

Figure 3.4 *For all the images, a 7 level image pyramid is fed into DeepPyramid. The output of the DeepPyramid are 7 sets of CNN feature maps and each set has 256 feature maps. After quantization, there are 1792 feature map scores in total.*

3.3 Dominant CNN feature channels

Dominant CNN feature channel ("dominant channel" in short) is the essential concept in our framework. In our hypothesis, an object may present distinct features in different poses and viewpoints. Therefore, there also should be some convolutional feature channels which are supposed to respond to, at least but not limited to, those significant features in those poses or viewpoints. Take bicycle as an example, it is obvious that there are significant differences between the viewpoint from pure front view ('T' shape) and pure side view (two round wheels). For the other poses between them, they contain both front view and side view features in different levels depending on the angle of the viewpoint. We define those distinct feature sensitive channels as dominant channels. It is exactly what we observed in Figure 3.2: channel 246 dominants the side view of the bicycle and has the very limited sensitivity to the front view of the bicycle while channel 186 does the opposite.

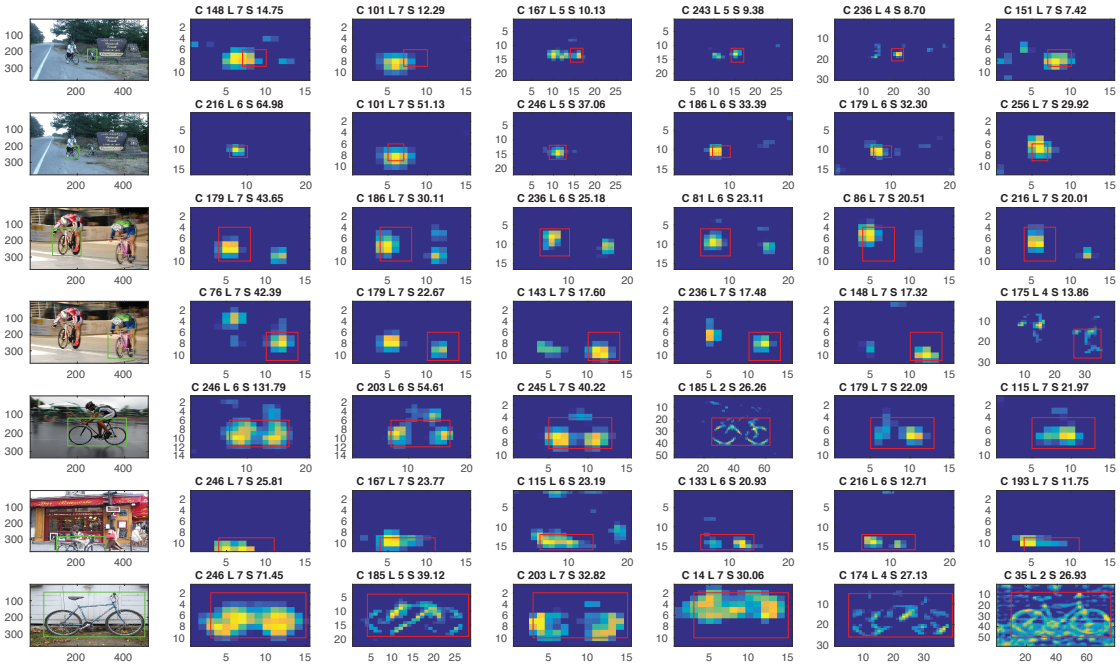

Figure 3.5 *Feature maps sorted by channel scores. S indicates the channel score. C and L shows from which channel and in which level the feature map is.*

The number of dominant channels defines the dimensions of the feature vector which will be used in the further clustering process. It is the same idea of the vocabulary vector in BoW model. In the example given above, if the pure front view sensitive channel 186 and pure side view sensitive channel 246 are the dominant channels, then the clustering will be performed in a two dimension space and each channel will be one of the axes. Mining the dominant channels is not only about to find the distinct feature channels, but also helps to reduce the dimensions significantly.

We proposed a method to mine the dominant channels based on the channel scores and also take the frequency of top score channels into account. Firstly, we find the top N most frequent channels in all the top K highest channel scores over the image set. Secondly, we sum the channel scores of all the images and pick the top N highest score channels. Last, pick those in both of the top frequency and top score channels as the dominant channels

3.3.1 Top frequent channels

Once we calculated the channel scores for all the images, the first step is to sort channel scores of each image in descending order. Then we pick the top K highest score channels of each image to calculate the histogram. K is a parameter to tune. The principle is to only focus on those most sensitive channels (with high channel scores). In this thesis, we decided to use the top one highest score channels, because the idea is to find the channels which respond to the most significant feature of the input image. As described in section 3.2, the most significant features produce the highest scores.

Still using the bicycle example, the histogram of top one highest score channels of all the samples are shown in Figure 3.6 in blue colour. The top 10 frequent channels are: [246 115 143 35 171 216 101 167 179 236]. It matches the description in Figure 3.2. Furthermore, from the frequency, it also matches the fact that in this image set, there are way more side view bicycle images than from other viewpoints.

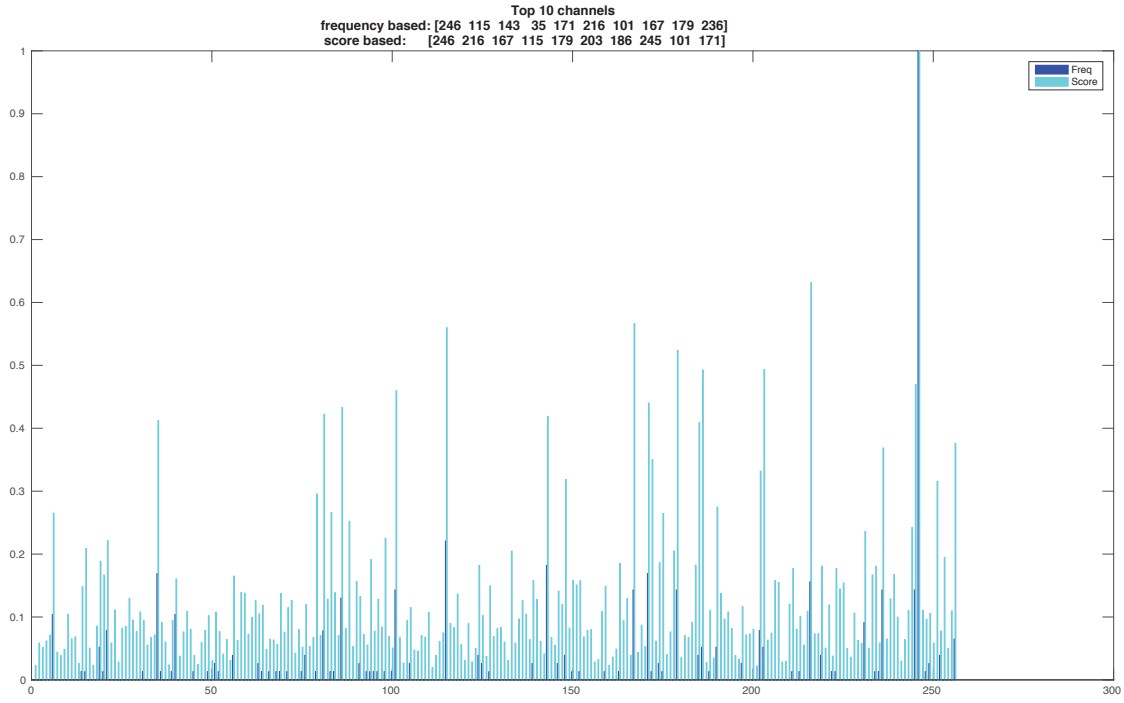

Figure 3.6 *The normalised frequency and score based channel histogram of images in bicycle category. The blue one is the histogram of the frequency of all the top one highest score channels of each image, the other one is the sum of channel scores of all the images.*

3.3.2 Top score channels

It is obvious that the higher the channel score is, the more significant the channel is. So the dominant channels should also be defined as those with highest channel scores. In order to do that, we first sum the feature map scores in the same channels (all 7 levels) of all the images, and then we sort the channel scores in descend order and pick the top ones. Figure 3.7 demonstrates the whole process. Each image has one row in the feature map scores.

Also using the bicycle example, the result shows in Figure 3.6. The top 10 highest score channels are: [246 216 167 115 179 203 186 245 101 171]. The first one is the same as in previous frequency based method and can find more common ones just in different order, such as channel 115, 171, 216, 101, 167.

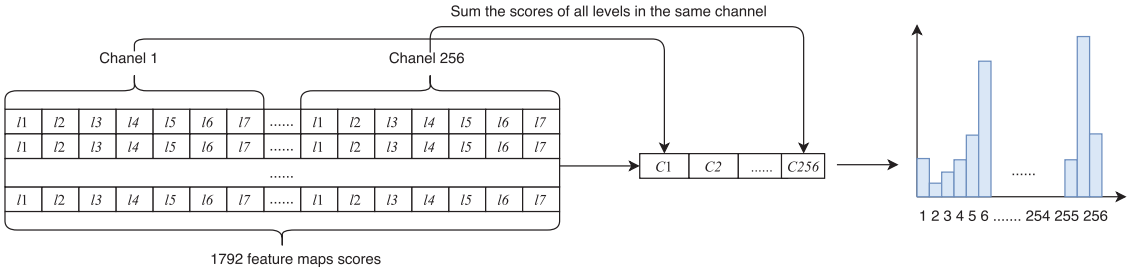

Figure 3.7 *Sum all the feature map scores in the same channel and pick the highest score channels as dominant channels. In this example, the dominant channels are channel 6 and channel 255, if we decide to pick the top two highest scores.*

3.3.3 Dominant channels

Dominant channels represent the most distinguished features so that they are supposed to be in both top score and high frequency channels. It is very straightforward to do it by picking the common channels in top N frequent channels and top N highest score channels mentioned previously.

Referring the bicycle example in section 3.3.1 and section 3.3.2, let's set $N = 7$, the dominant channels for the bicycle are [246 115 216]. Figure 3.8 is the visualization of those channels on five bicycle images.

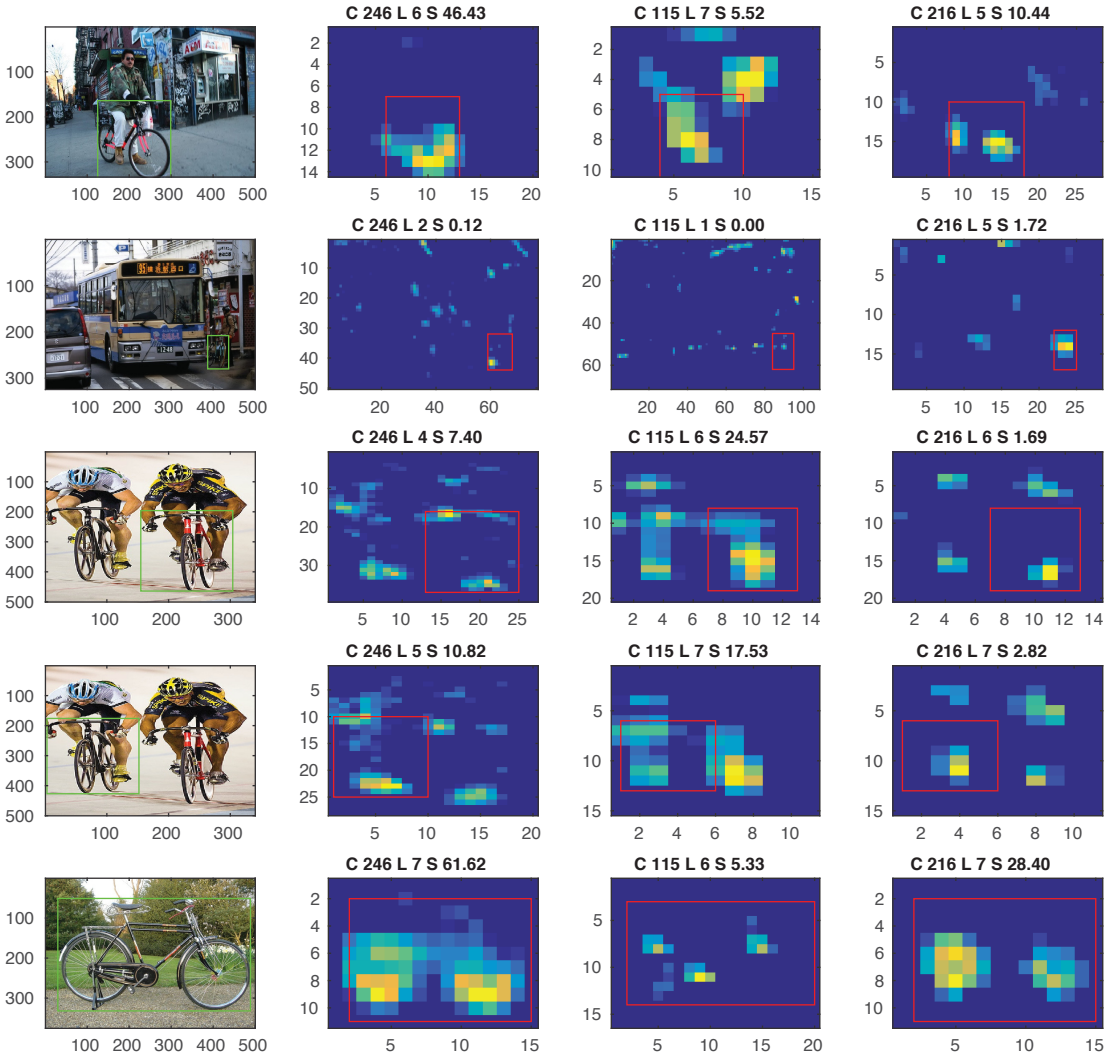

Figure 3.8 *C and S denote the channel and the channel score (sum of all levels of the channel). As for each channel there are 7 levels, here we only pick the level with highest channel score. The yellow pixels represent high response and blue ones are for low response.*

3.4 Clustering

The dominant CNN feature channels produce the feature vector for the clustering. As there are only a few dimensions in the feature vector, it is reasonable to apply the cluster algorithms such as K-means or Fuzzy c-means. Although they all need to define the number of clusters before hand, in real life, there are only limited number of potential poses or viewpoints. Therefore, we can manually set and test different cluster numbers. In the work [7, 10], 3 components configuration gains the best

performance in their experiments.

3.4.1 Feature vector

To generate feature vector of each image is the last step before clustering. The procedure is very straightforward as shown in Figure 3.9. With the dominant channels, we go through all the channel scores of each image and only pick the dominant channel scores to form its feature vector.

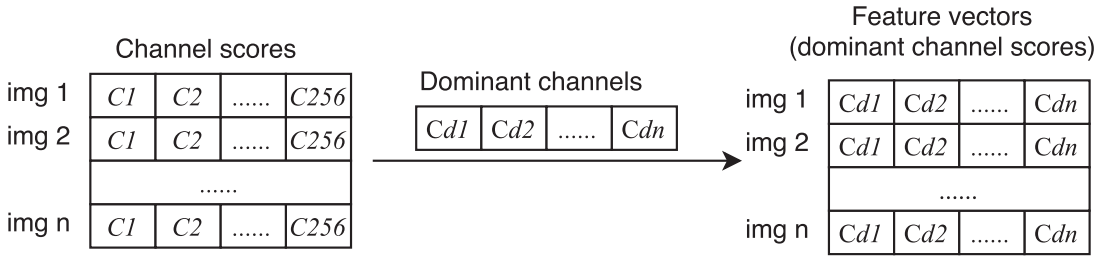

Figure 3.9 *For each image, the channel scores of every channel in all the 7 levels are summed up as a single channel score. Given the dominant channels, pick the corresponding channel scores to form the final feature vectors.*

3.4.2 K-means and Fuzzy c-means implementation

MATLAB provides function $idx = kmeans(X, k, ...)$ for K-means clustering. It receives two arguments: n-by-p data matrix X and the number of clusters k. The return value is a n-by-1 vector contains the cluster index for each element. By default, the distance is defined as squared Euclidean distance and the centre initialization uses k-means++ algorithm [1] which outperforms other algorithms in the running time and the quality of the final cluster results.

As the same for K-means, MATLAB also has the built-in function $[centers, U] = fcm(data, Nc)$ for Fuzzy c-means clustering, in which the $data$ is a n-by-d matrix for n elements with d dimensions, Nc is the number of clusters, the return values are the cluster $centers$ and membership U for each element in all the clusters.

Although both clustering methods are quite similar, they do generate way different cluster results in many cases.

4. RESULTS AND ANALYSIS

It is very subjective to classify objects into different poses or viewpoints, and there is no ground truth to compare the clustering results from different algorithms or different configurations in the same method directly. Nevertheless, we trained and tested the DPM detector [7] on VOC 2007 image set as a benchmark. Then we retrain it by replacing its original aspect ratio initialization method with our dominant channel clustering approach and test it again. In our approach, we performed both K-means and Fuzzy c-means clustering. The dominant channels are picked from top 10 ($N = 10$) and top 5 ($N = 5$) most frequent and highest score channels. In the DPM settings, we also use 3 components in all of our training process as it is proved to be the optimized option in [7]. The detail results are listed in Table 4.1.

Table 4.1 *Detection results, AP in percentage, for different subcategory mining methods on VOC 2007 dataset. The winner is in* ***bold*** *font.*

	N	plane	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	motor	person	plant	sheep	sofa	train	tv	mAP
DPM [7]		**31.5**	57.8	**11.0**	**16.7**	25.4	**50.9**	53.0	17.9	20.8	25.5	27.2	**12.1**	**56.0**	**47.8**	**43.2**	**13.4**	22.6	**35.5**	**44.5**	40.2	**32.6**
DC fcm	10	29.0	57.5	10.5	13.4	**28.4**	41.9	51.9	21.0	20.5	23.8	32.9	**12.1**	53.0	45.5	38.1	12.8	**26.0**	31.8	41.6	**43.1**	31.7
DC km	10	29.8	57.3	10.4	13.2	26.3	42.4	53.1	18.5	21.2	23.6	24.7	4.8	49.9	43.2	39.5	12.3	17.5	24.2	34.2	41.2	29.4
DC fcm	5	25.8	**58.3**	10.2	15.7	26.2	41.9	**53.3**	19.5	20.5	24.5	**33.3**	6.4	55.7	43.5	38.5	12.1	18.3	32.8	41.7	42.2	31.0
DC km	5	28.9	56.3	3.8	13.3	24.1	42.3	52.2	**22.3**	**21.5**	**27.8**	22.8	6.8	53.8	44.3	39.0	**13.4**	20.5	30.4	40.9	43.0	30.4

It shows that our framework doesn't perform well especially in categories of boat, bus, person, sofa and train. However, we do achieve notable improvements in categories of bottle, cat, table, sheep and TV monitor.

The following analyses are based on our best average AP performance approach: Dominant Channels with Fuzzy c-means (DC fcm and N = 10, mAP = 31.7).

Figure 4.1 shows the results of our clustering method and aspect ratio based method on the same images from boat category. There are three subcategories of boats: sailboat, yacht and normal boat. They have the very distinct appearances that the sailboat is tall, the yacht has a roof and the normal boat is very low and long. Thus, in the boat category, the subcategory problem is not only about the viewpoints and poses, but also involving inner-calss variety. Therefore, the aspect ratio method

works very well to separate those three subcategories because different types of the boat have the distinct aspect ratio in appearance. In the other side, our method didn't handle the inner-class variety nicely.

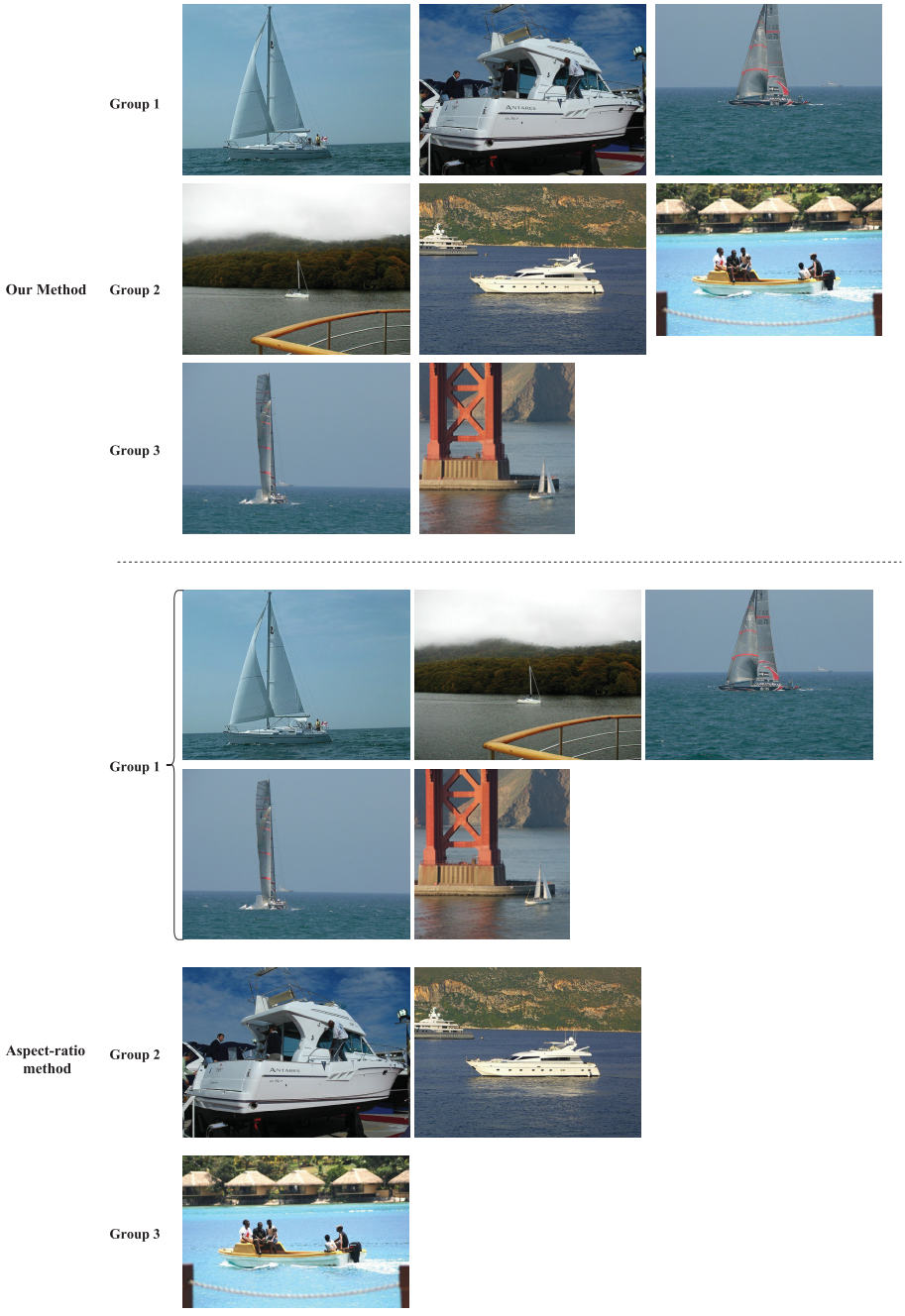

Figure 4.1 *Example of subcategory clustering results for boat images.*

Further examining the visualization of the dominant CNN feature channels for the boat, we found our method could do better by choosing the dominant CNN feature

channels more wisely. As illustrated in Figure 4.2, we can see obviously that channel 225 is highly sensitive to the sailboat, particularly to the sail part, while the channel 70 is responding to the overall shape of the yacht. From the above observation, it is somehow managed to category the boats into different types by only using channel 225 and 70. However, even we can subcategorize different types of the boat, there are still different viewpoints and poses in each type of them, such as the front view of yacht and the side view of yacht. It is different from the bike case in which there is no huge inner-class variety but only encountering pose or viewpoint variety.

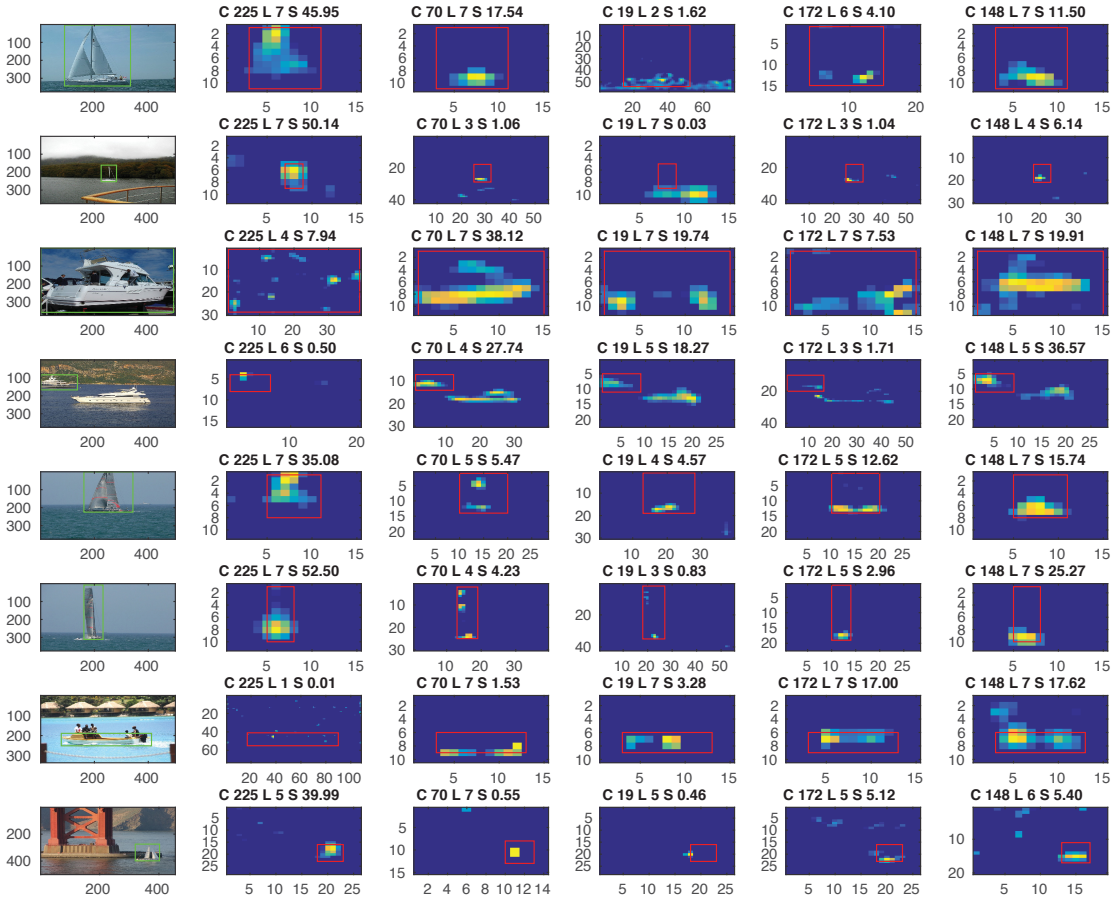

Figure 4.2 *Visualization of the dominant channels for boats.*

The biggest performance gap between our method and the original one happened in the category of bus. Figure 4.3 gives an example of clustering results on bus category. Same with the boat case, the aspect ratio method is able to separate the side view and back view of the bus because of the obvious aspect ratio difference. Our result looks like randomly picking.

Figure 4.3 *Example of subcategory clustering results for bus images.*

Looking at the visualization of the dominant CNN feature channels for the bus, Figure 4.4 reveals that our score strategy doesn't well address the most distinguished feature of the bus in different viewpoints and poses. Our score strategy trends to find the large size of key features of the object, for example, the wheel of the bike, the sail of the sailboat, etc. But in the bus scenario, the key feature of the front face is the headlights which only occupy a small area of the scene. There is no channel found to associate with such a small but highly distinguished feature in our dominant channels. They all tend to learn the overall shape of the object which, in most of the cases, are rectangles. In such sense, the aspect-ratio method could be

better to separate the front view (square) and the side view (rectangle) of the bus.

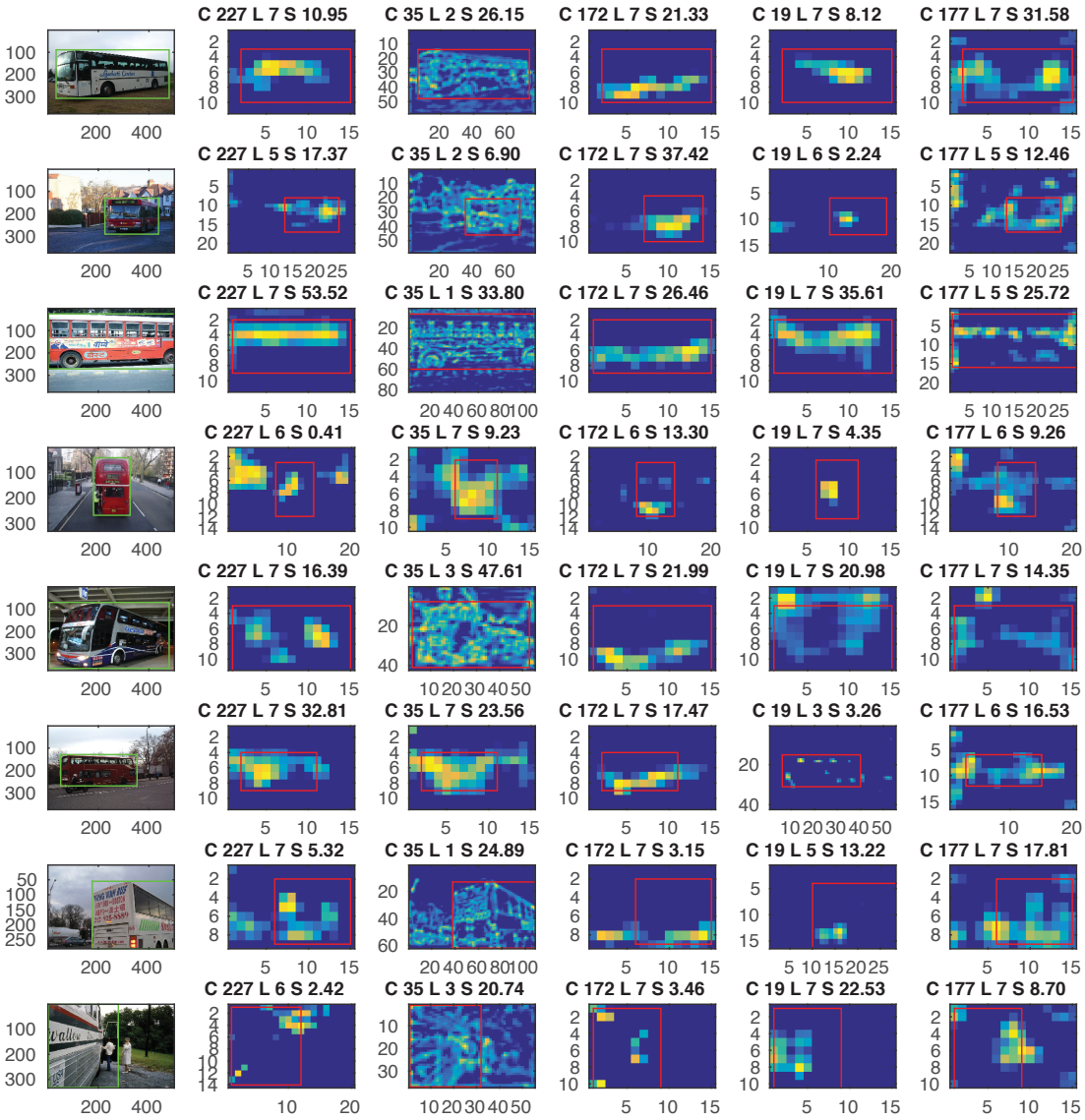

Figure 4.4 *Visualization of the dominant channels for buses.*

In other categories, our approach could outperform the aspect ratio based method, especially in the cat category in which every experiment is better than the original one. Given the cat images in Figure 4.5, it is even hard for the human to subcategorize them. The cat category is a typical case involving both viewpoints and poses. For example, the last image in both group 1 and group 2 of our method, those two cats are all in the same pose of lying on their stomach, but it is in front view for the one in group 1 while the one in group 2 is in a bird view. Although it is a highly

subjective task to subcategorize the cat images rather than the boat images, in our method, the group 2 tends to have the bird view of the cats and images in group 1 look like containing cat head with clear ears and eyes. In the contrast, the result from the aspect ratio based method didn't show any clue of the difference between groups.

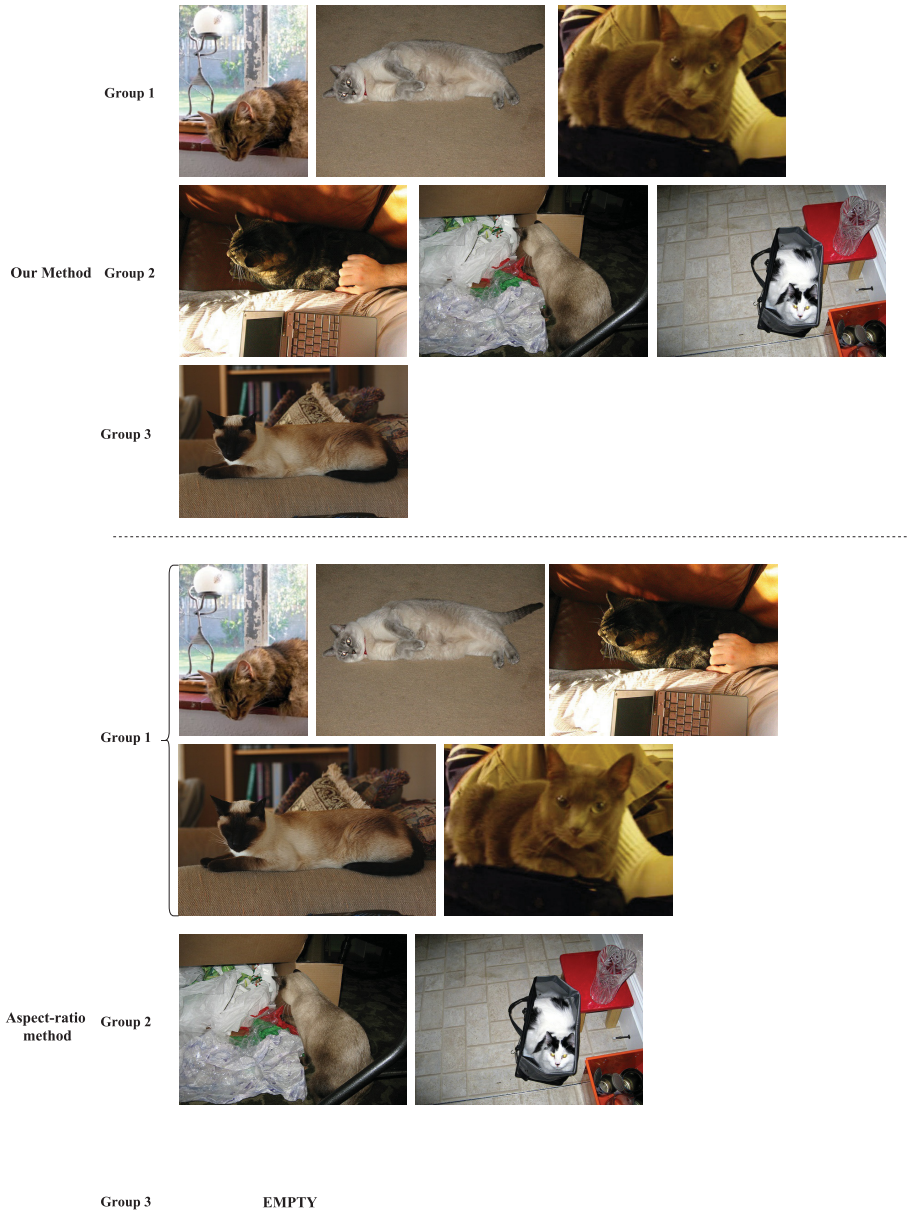

Figure 4.5 *Example of subcategory clustering results for cat images.*

From Figure 4.6, it is also hard to interpret which particular feature of the cat the dominant CNN feature channels are learning. It makes sense because there are many

viewpoints and especially lots of possible poses a cat can perform, not to mention about the variety of colour and patterns they can have.

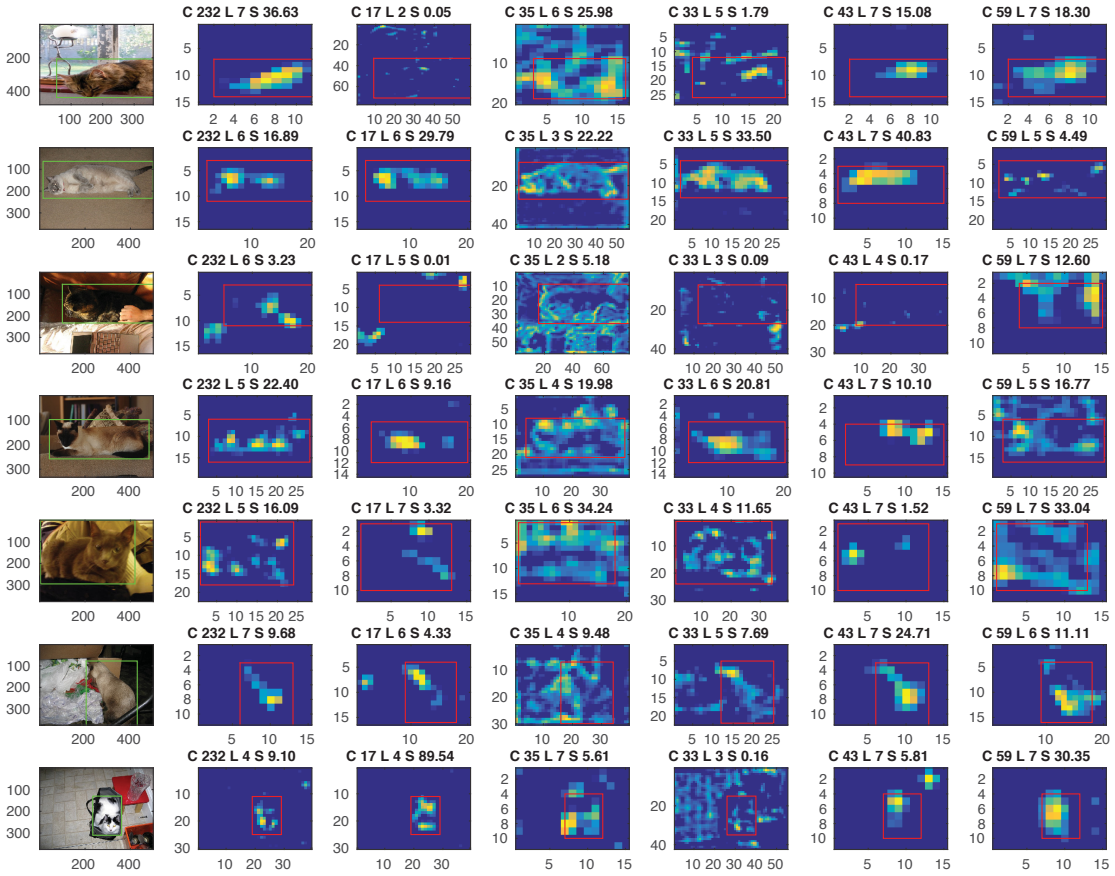

Figure 4.6 *Visualization of the dominant channels for cats.*

The other difference between our approach and the aspect-ratio method is that the aspect-ratio method tends to split the images evenly regardless of the number of subcategories, while in our clustering process, the number of images in each category can be unevenly generated, which could result lacking of training samples in some of the subcategories.

In short, the advantage of our method is to categorize the images based on features which can achieve better performance in the categories with more poses and viewpoints varieties such as cat, cow, sheep. However, the limitations are also notable: 1) our channel score strategy could not reflect the distinct features in small size, such as headlight of the bus. 2) the strategy of picking dominant channels also needs to improve.

5. CONCLUSIONS

The result carried out from this thesis revealed the potential application on CNN features for image subcategorizing. Our hypothesis on the concept of dominant channels is proved to be a useful discovery for further investigation. Furthermore, our experiment results show Fuzzy c-means is a better choice than K-means in our framework.

Although our framework doesn't outperform the original aspect ratio based DPM [7] in the average AP through all 20 categories, we still gained notable performance improvement in several individual categories. In other words, there are still potentials to explore the CNN features. For example, in our method, there are usually more than one dominant channels respond to the same features, such as channel 245 and 246 both response to the wheels in bicycle example. As we mentioned in the hypothesis, the idea is to find the most distinguished features in different viewpoints or poses by mining the dominant channels. If there are more than one channels highly sensitive to the same viewpoint or pose, we can try to merge them together rather than treat them in different dimensions which can reduce the feature dimension one step further and could be more accurate in representing those most significant features. Another idea is to apply some text mining techniques such as TF-IDF as our method is very similar to BoW model so we can borrow some ideas from it. One more tip is to try some other latest successful CNNs architectures, such as GoogLeNet, Microsoft ResNet and R-CNNs. The last, but not the least, point is to improve the channel score model to represent the channel sensitivity in a more accurate way.

In summary, we explored one useful application on CNN feature maps and present a framework successfully outperformed the existing image classification and detection system in some particular categories. And we also proposed the potential approaches to improve our work in the future.

BIBLIOGRAPHY

[1] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2007.

[2] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, 1981.

[3] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[4] S. K. Divvala, A. A. Efros, and M. Hebert, "How important are "deformable parts" in the deformable parts model," in *European Conference on Computer Vision (ECCV) Workshop on Parts and Attributes*, 2012.

[5] J. Dong, W. Xia, Q. Chen, J. Feng, Z. Huang, and S. Yan, "Subcategory-aware object classification," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[6] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, 2015.

[7] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[8] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2010.

[9] M. A. Fischler and R. A. Elschlager, "The representation and matching of pictorial structures," *IEEE Trans. Comput.*, 1973.

[10] R. Girshick, F. Iandola, T. Darrell, and J. Malik, "Deformable part models are convolutional neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[11] D. Hall and P. Perona, "Fine-grained classification of pedestrians in video: Benchmark and state of the art," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[12] H. Hattori, V. N. Boddeti, K. M. Kitani, and T. Kanade, "Learning scene-specific pedestrian detectors without real data," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[13] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, 1989.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012.

[15] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[16] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, 1998.

[17] F.-F. Li and P. Perona, "A bayesian hierarchical model for learning natural scene categories," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[18] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua, "A convolutional neural network cascade for face detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[19] S. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory, Volume 28*, 2006.

[20] H. Qin, J. Yan, X. Li, and X. Hu, "Joint training of cascaded CNN for face detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[21] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, 2015.

[22] J. Schmidhuber, "Multi-column deep neural networks for image classification," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[23] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong, "Locality-constrained linear coding for image classification," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[24] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European Conference on Computer Vision (ECCV)*, 2014.

[25] S. Zhang, R. Benenson, M. Omran, J. H. Hosang, and B. Schiele, "How far are we from solving pedestrian detection?" in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[26] X. Zhu, C. Vondrick, D. Ramanan, and C. C. Fowlkes, "Do we need more training data or better models for object detection?" in *British Machine Vision Conference (BMVC)*, 2012.